	L #	Hits	Search Text	DBs
1	L1	93761	(speculative$3 sequen$4 order) near10 (instruction execut$3)	USPAT; US-PGPUB
2	L2	1692	(indicat$3 flag tag bit field mode) near10 (allow$3 permit$4 enabl$3 disabl$3 prohibit$3 inhibit$3 activ$5 deactiv$5) near20 1	USPAT; US-PGPUB
3	L3	445	((speculative$3 sequen$4 order) and (instruction execut$3)).ab,ti. and 2	USPAT; US-PGPUB
4	L8	95938	(speculative$3 sequen$4 order predict$3) near10 (instruction execut$3)	USPAT; US-PGPUB
5	L9	1793	(indicat$3 flag tag bit field mode) near10 (allow$3 permit$4 enabl$3 disabl$3 prohibit$3 inhibit$3 activ$5 deactiv$5) near20 8	USPAT; US-PGPUB
6	L10	527	((speculative$3 sequen$4 order predict$3) and (instruction execut$3)).ab,ti. and 9	USPAT; US-PGPUB
7	L11	31579	(speculative$3 sequen$4 order predict$3) near10 (instruction execut$3)	EPO; JPO; DERWENT; IBM_TDB
8	L12	157	(indicat$3 flag tag bit field mode) near10 (allow$3 permit$4 enabl$3 disabl$3 prohibit$3 inhibit$3 activ$5 deactiv$5) near20 11	EPO; JPO; DERWENT; IBM_TDB
9	L13	62	9 near10 speculative$3	USPAT; US-PGPUB
10	L19	95947	(speculat$6 sequen$4 order predict$3) near10 (instruction execut$3)	USPAT; US-PGPUB
11	L22	31612	(speculat$6 sequen$4 order predict$3) near10 (instruction execut$3)	EPO; JPO; DERWENT; IBM_TDB
12	L33	1734	(indicat$3 flag tag bit field mode) near10 ((allow$3 permit$4 enabl$3 disabl$3 prohibit$3 inhibit$3 activ$5 deactiv$5) near10 19)	USPAT; US-PGPUB
13	L34	563	((speculat$6 sequen$4 order predict$3 load) and (instruction execut$3)).ab,ti. and 33	USPAT; US-PGPUB
14	L35	25	(indicat$3 flag tag bit field mode) near10 ((allow$3 permit$4 enabl$3 disabl$3 prohibit$3 inhibit$3 activ$5 deactiv$5) near10 22) not 12	EPO; JPO; DERWENT; IBM_TDB
15	L36	56467	(speculat$6 order predict$3) near10 (instruction execut$3)	USPAT; US-PGPUB
16	L37	273	(indicat$3 flag tag bit field mode) near10 ((allow$3 permit$4 enabl$3 disabl$3 prohibit$3 inhibit$3 activ$5 deactiv$5) near10 36) and 34 not 13	USPAT; US-PGPUB
17	L38	243	34 not (37 13)	USPAT; US-PGPUB

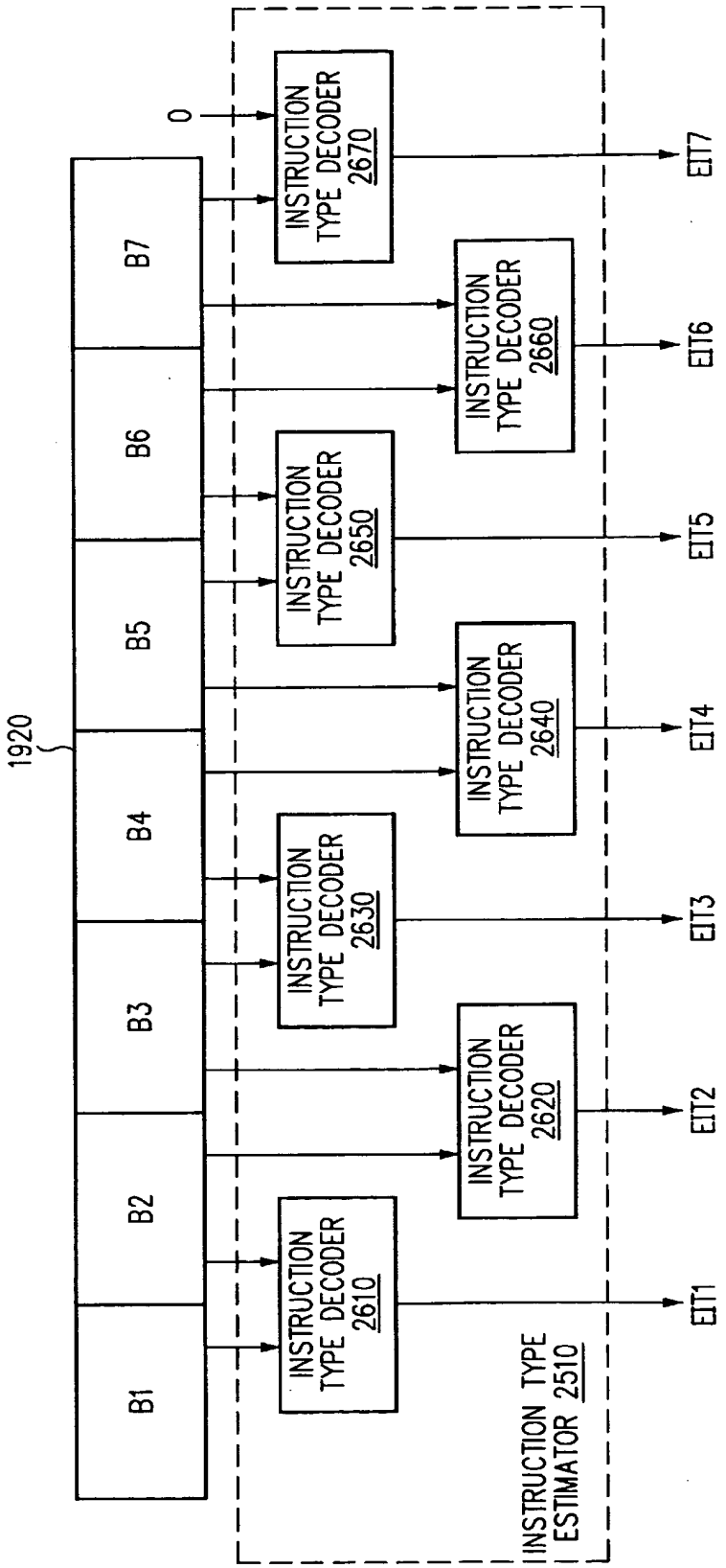
1920
B1
B2
B3
B4
B5
B6
B7
0
INSTRUCTION TYPE DECODER 2610
INSTRUCTION TYPE DECODER 2620
INSTRUCTION TYPE DECODER 2630
INSTRUCTION TYPE DECODER 2640
INSTRUCTION TYPE DECODER 2650
INSTRUCTION TYPE DECODER 2660
INSTRUCTION TYPE DECODER 2670
EIT1
EIT2
EIT3
EIT4
EIT5
EIT6
EIT7
INSTRUCTION TYPE ESTIMATOR 2510

FIG. 26

	Document ID	U	Title	Current OR
1	JP 20032 08335 A	☐	WINDOW MENU TEST SYSTEM	
2	JP 20010 34167 A	☐	ARITHMETIC UNIT AND CRYPTOGRAM PROCESSOR	
3	JP 41134 7892 A	☐	MIXED FLOW PRODUCTION SYSTEM	
4	JP 10154 073 A	☐	DEVICE AND METHOD FOR MANAGING DATA DEPENDENCY	
5	JP 08251 017 A	☐	TEST CONTROL CIRCUIT FOR SEQUENTIAL CIRCUIT, AND TEST METHOD THEREOF	
6	JP 07311 660 A	☐	METHOD FOR PROCESSING DATA IN SOFTWARE CONTROL SYSTEM FOR EARLY DETECTING OBSTACLE	
7	JP 07302 189 A	☐	ADDER AND ADDING METHOD	
8	JP 07078 088 A	☐	DEVICE FOR PROCESSING PLURAL TIMES OF INTERRUPTION AT PROCESSOR SYSTEM, METHOD FOR REACTING TO INTERRUPTIONS FROM PLURAL INTERRUPTION TRIGGERS AT PROCESSOR SYSTEM, METHOD AND DEVICE FOR SIMULTANEOUS T	
9	JP 06242 949 A	☐	QUEUE MANAGEMENT TYPE INSTRUCTION CACHE	
10	JP 04096 496 A	☐	CONTROL DATA TRANSMISSION SYSTEM	
11	JP 02142 272 A	☐	FACSIMILE EQUIPMENT	
12	JP 01036 399 A	☐	VOICE ALARM OUTPUT DEVICE FOR OUTBOARD MOTOR	
13	WO 20040 01586 A1	☐	DEVICE AND METHOD FOR PROCESSING A SEQUENCE OF JUMP INSTRUCTIONS	
14	EP 10963 71 A1	☐	A method and apparatus for prefetching instructions	
15	EP 63780 2 A2	☐	Interrupt vector method and apparatus.	
16	US 20030 08875 9 A	☐	Scratch value identification method in e.g. personal computer, involves setting tags associated with data areas of processor to indicate that data areas hold scratch values	
17	US 20020 13823 6 A	☐	Execution result prediction processor increases priority of priority table based on whether predicted value generated using history table and execution result of instruction being executed currently, are similar	
18	US 20010 02095 2 A	☐	Script producing apparatus for proof image producing system, has script producing section for producing script representative of execution order which is recognized by processor based on execution diagram	
19	EP 10469 83 A	☐	Very long instruction word processor executes sub-instructions in parallel and has an exchange system to exchange sub-instructions prior to execution	
20	US 59999 69 A	☐	Message transfer protocol executing method in interrupt handling system connected to network with hard and soft emulated digital modules	

U.S. PATENT DOCUMENTS

5,247,644	9/1993	Johnson et al. .	
5,295,252	3/1994	Torii et al. .	
5,371,864	12/1994	Chuang	712/210
5,379,396	1/1995	Gochman et al. .	
5,448,746	9/1995	Eickemeyer et al.	712/210
5,463,759	10/1995	Ghosh et al. .	
5,485,572	1/1996	Overley .	
5,500,942	3/1996	Eickemeyer et al.	712/210
5,517,660	5/1996	Rosich .	
5,535,350	7/1996	Maemura .	
5,559,986	9/1996	Alpert et al. .	
5,581,774	12/1996	Yoshitake et al.	712/210
5,584,009	12/1996	Garibay, Jr. et al. .	
5,603,006	2/1997	Satake et al. .	
5,625,784	4/1997	Purcell	712/210
5,634,027	5/1997	Saito .	
5,636,362	6/1997	Stone et al. .	
5,687,336	11/1997	Shen et al. .	
5,757,295	5/1998	Bakhmutsky	341/67
5,784,553	7/1998	Kolawa et al. .	
5,961,638	10/1999	Tran	712/239
5,973,628	10/1999	McDonnell et al.	341/67
6,008,745	12/1999	Zandi et al.	341/67

	Document ID	U	Title	Current OR
21	GB 2275805 A	☐	Electronic cash register - has expanded macro key functions set so that only operator having manager level to operate register at predetermined time.	
22	EP 565712 B	☐	Network structure for parallel processing computer programs - uses large number of highly efficient virtual processors derived from large memory banks in the core of the network	
23	EP 547245 B	☐	Image processing method for industrial visual sensor in robot controller - fetches image data from auxiliary memory using flags sequentially in frame memory when production line stopped, executes image processing program and allows failures to be analysed from monitor display	
24	US 4170832 A	☐	Interactive teaching machine with video recorded material - has recording of events leading to several possible choices and buttons selecting playback of consequences of each choice	
25	GB 1446535 A	☐	Semi-automatic wiring machine - has target arrangement indicating each terminal in turn to be wired	

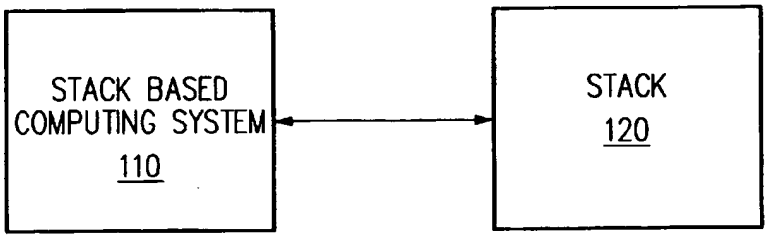

FIG. 1

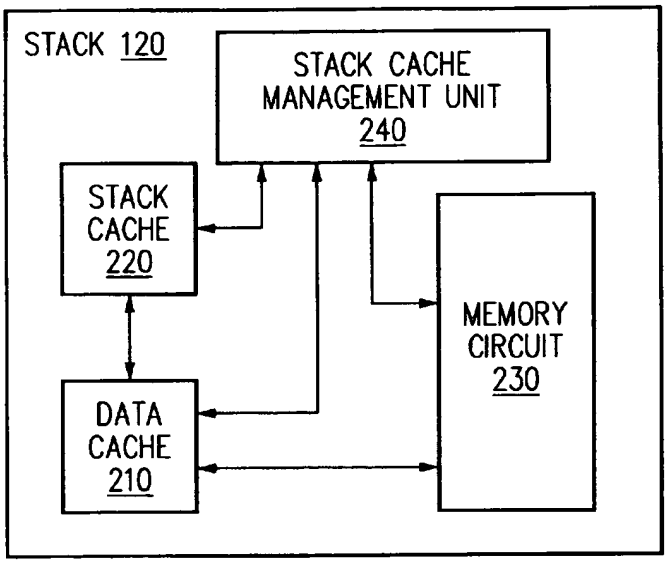

FIG. 2

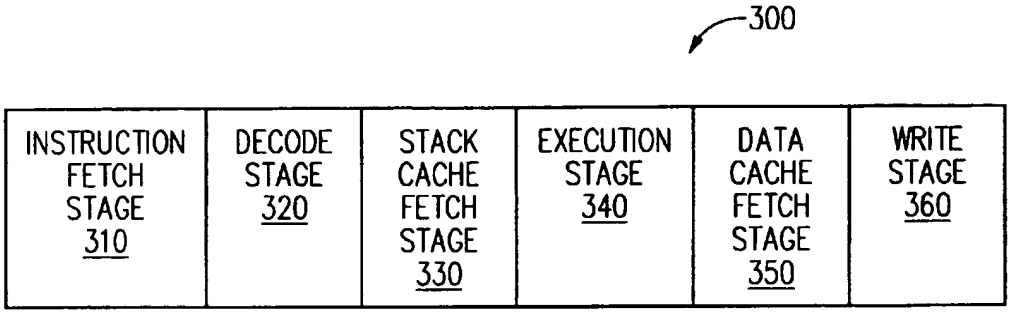

FIG. 3

	Document ID	U	Title	Current OR
1	US 20040054872 A1	☐	High-performance, superscalar-based computer system with out-of-order intstruction execution	712/206
2	US 20040049660 A1	☒	Method and apparatus for clearing hazards using jump instructions	712/219
3	US 20040003205 A1	☒	Apparatus and method for executing instructions	712/215
4	US 20040002913 A1	☒	Online trade aggregating system	705/37
5	US 20030229548 A1	☒	Cash home-delivery system and cash home-delivery method	705/26
6	US 20030217251 A1	☒	Prediction of load-store dependencies in a processing agent	712/225
7	US 20030208665 A1	☒	Reducing data speculation penalty with early cache hit/miss prediction	711/169
8	US 20030204705 A1	☒	Prediction of branch instructions in a data processing apparatus	712/207
9	US 20030182543 A1	☒	TRAINING LINE PREDICTOR FOR BRANCH TARGETS	712/237
10	US 20030182287 A1	☒	Interface for an electronic spreadsheet and a database management system	707/10
11	US 20030182203 A1	☒	Method for supporting shipment of virtual shopping mall	705/26
12	US 20030182050 A1	☒	Apparatus and method for diagnosis of vehicular system	701/114
13	US 20030177086 A1	☒	Integrated order pre-matching system	705/37
14	US 20030163670 A1	☒	Re-encoding illegal OP codes into a single illegal OP code to accommodate the extra bits associated with pre-decoded instructions	712/209
15	US 20030159019 A1	☒	Prediction of instructions in a data processing apparatus	712/207
16	US 20030154362 A1	☒	Method and system for safe data dependency collapsing based on control-flow speculation	712/216
17	US 20030140245 A1	☒	Secure mode for processors supporting MMU and interrupts	713/200

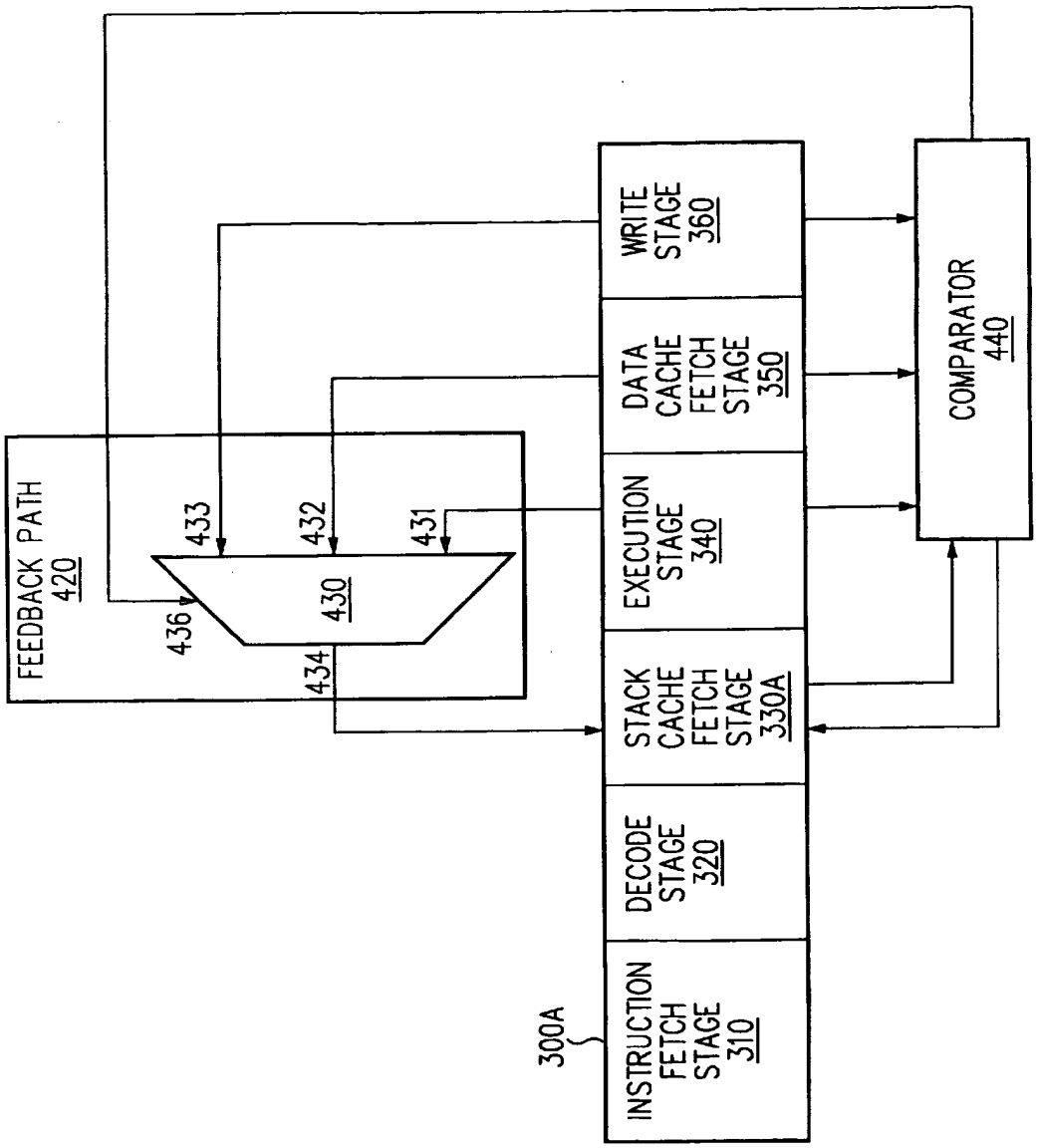
INSTRUCTION FETCH STAGE 310
DECODE STAGE 320
STACK CACHE FETCH STAGE 330A
EXECUTION STAGE 340
DATA CACHE FETCH STAGE 350
WRITE STAGE 360
COMPARATOR 440
FEEDBACK PATH 420
430
431
432
433
434
436
300A

FIG. 4

	Document ID	U	Title	Current OR
18	US 20030140244 A1	☒	Secure mode for processors supporting MMU	713/200
19	US 20030140205 A1	☒	Secure mode for processors supporting interrupts	711/163
20	US 20030135722 A1	☒	Speculative load instructions with retry	712/235
21	US 20030126066 A1	☒	Electronic communication network ranking for automated market system	705/37
22	US 20030126065 A1	☒	Order price threshold for automated market system	705/37
23	US 20030079113 A1	☒	High-performance, superscalar-based computer system with out-of-order instruction execution	712/205
24	US 20030070060 A1	☒	High-performance, superscalar-based computer system with out-of-order instruction execution	712/23
25	US 20030056087 A1	☒	High-performance, superscalar-based computer system with out-of-order instruction execution	712/207
26	US 20030056086 A1	☒	High-performance, superscalar-based computer system with out-of-order instruction execution	712/207
27	US 20030040984 A1	☒	Image processing apparatus, method of placing order for expendables of image processing apparatus, storage medium, and program	705/27
28	US 20030038808 A1	☒	Method, apparatus and article of manufacture for a sequencer in a transform/lighting module capable of processing multiple independent execution threads	345/506
29	US 20030028446 A1	☒	Web-enabled method and system for searching correct model data indicative of a porduct to be purchased online	705/27
30	US 20030024518 A1	☒	Computer system, particularly for simulation of human perception via sense organs	124/50
31	US 20030020720 A1	☒	Method, apparatus and article of manufacture for a sequencer in a transform/lighting module capable of processing multiple independent execution threads	345/506
32	US 20020194456 A1	☒	System and method for register renaming	712/217
33	US 20020157000 A1	☒	Software hint to improve the branch target prediction accuracy	712/239
34	US 20020152368 A1	☒	Processor with value predictor	712/226

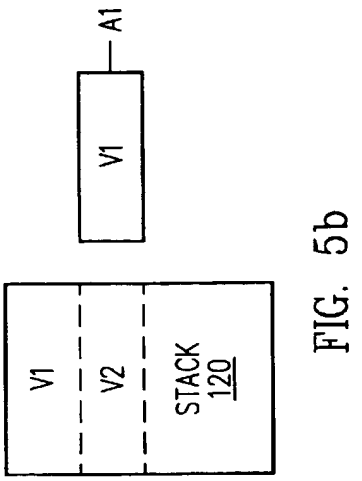

FIG. 5b

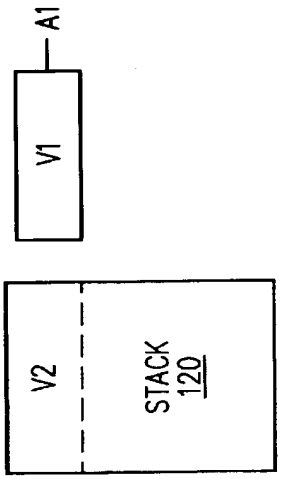

FIG. 6b

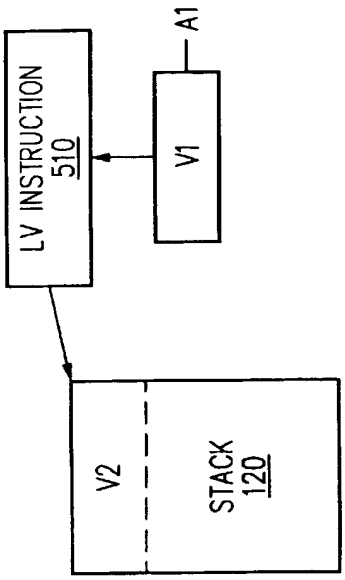

FIG. 5a

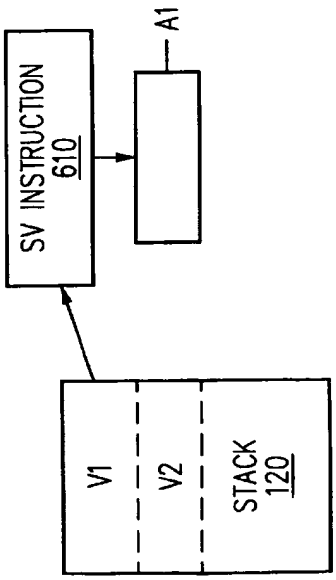

FIG. 6a

	Document ID	U	Title	Current OR
35	US 20020144098 A1	☒	Register rotation prediction and precomputation	712/221
36	US 20020138236 A1	☒	Processor having execution result prediction function for instruction	702/186
37	US 20020107781 A1	☒	Compound order handling in an anonymoustrading system	705/37
38	US 20020095438 A1	☒	Proposed syntax for a synchronized commands execution	715/500.1
39	US 20020091913 A1	☒	Re-order buffer managing method and processor	712/218
40	US 20020083312 A1	☒	Branch Prediction apparatus and process for restoring replaced branch history for use in future branch predictions for an executing program	712/240
41	US 20020083300 A1	☒	System and method for register renaming	712/203
42	US 20020073357 A1	☒	Multiprocessor with pair-wise high reliability mode, and method therefore	714/19
43	US 20020073016 A1	☒	Order execution processing for automated market system	705/37
44	US 20020049961 A1	☒	Rule-based personalization framework	717/127
45	US 20020032841 A1	☒	Cache update method and cache update control system employing non-blocking type cache	711/133
46	US 20020029328 A1	☒	High-performance, superscalar-based computer system with out-of-order instruction execution	712/23
47	US 20020016903 A1	☒	High-performance, superscalar-based computer system with out-of-order instruction execution and concurrent results distribution	712/23
48	US 20010020952 A1	☒	Script producing apparatus and script producing program storage medium	345/704
49	US 20010011343 A1	☒	System and method for register renaming	712/217
50	US 20010007125 A1	☒	Computer system with debug facility	712/226
51	US 20010004755 A1	☒	MECHANISM FOR FREEING REGISTERS ON PROCESSORS THAT PERFORM DYNAMIC OUT-OF-ORDER EXECUTION OF INSTRUCTIONS USING RENAMING REGISTERS	712/217

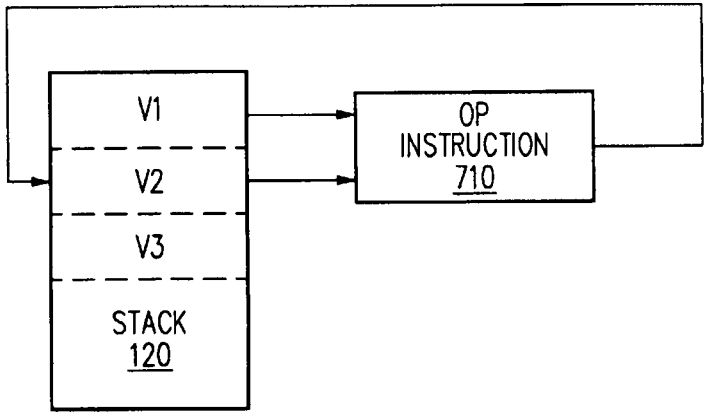

FIG. 7a

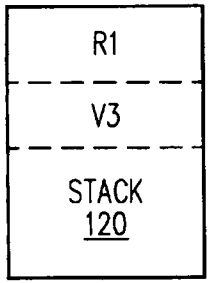

FIG. 7b

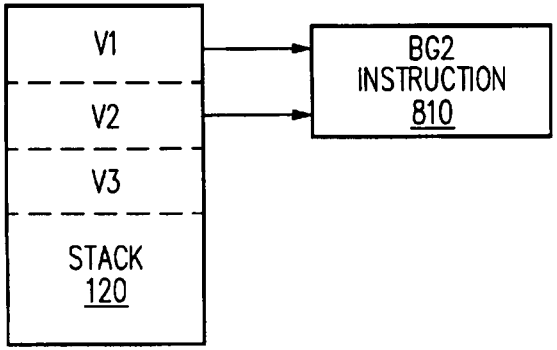

FIG. 8a

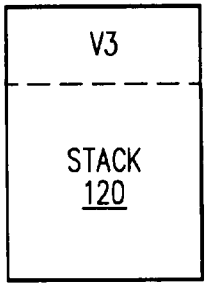

FIG. 8b

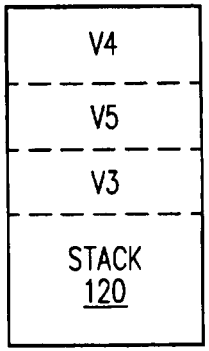

FIG. 8c

	Document ID	U	Title	Current OR
52	US 6697932 B1	☒	System and method for early resolution of low confidence branches and safe data cache accesses	712/216
53	US 6697789 B2	☒	Computer system, particularly for simulation of human perception via sense organs	706/12
54	US 6678638 B2	☒	Processor having execution result prediction function for instruction	702/186
55	US 6675298 B1	☒	Execution of instructions using op code lengths longer than standard op code lengths to encode data	713/190
56	US 6675291 B1	☒	Hardware device for parallel processing of any instruction within a set of instructions	712/236
57	US 6671798 B1	☒	Configurable branch prediction for a processor performing speculative execution	712/234
58	US 6650731 B1	☒	Simulator for simulating an intelligent network	379/15.01
59	US 6650330 B2	☒	Graphics system and method for processing multiple independent execution threads	345/506
60	US 6647490 B2	☒	Training line predictor for branch targets	712/233
61	US 6647485 B2	☒	High-performance, superscalar-based computer system with out-of-order instruction execution	712/23
62	US 6647463 B2	☒	Cache update method and cache update control system employing non-blocking type cache	711/118
63	US 6640315 B1	☒	Method and apparatus for enhancing instruction level parallelism	714/17
64	US 6636959 B1	☒	Predictor miss decoder updating line predictor storing instruction fetch address and alignment information upon instruction decode termination condition	712/204
65	US 6633970 B1	☒	Processor with registers storing committed/speculative data and a RAT state history recovery mechanism with retire pointer	712/217
66	US 6615188 B1	☒	Online trade aggregating system	705/37
67	US 6598154 B1	☒	Precoding branch instructions to reduce branch-penalty in pipelined processors	712/237
68	US 6573900 B1	☒	Method, apparatus and article of manufacture for a sequencer in a transform/lighting module capable of processing multiple independent execution threads	345/537
69	US H002064 H	☒	Automated fixed income trading	705/37
70	US 6557095 B1	☒	Scheduling operations using a dependency matrix	712/216
71	US 6546478 B1	☒	Line predictor entry with location pointers and control information for corresponding instructions in a cache line	712/204
72	US 6526502 B1	☒	Apparatus and method for speculatively updating global branch history with branch prediction prior to resolution of branch outcome	712/239
73	US 6516405 B1	☒	Method and system for safe data dependency collapsing based on control-flow speculation	712/216
74	US 6502188 B1	☒	Dynamic classification of conditional branches in global history branch prediction	712/234

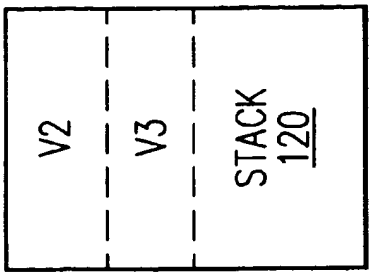

FIG. 9b

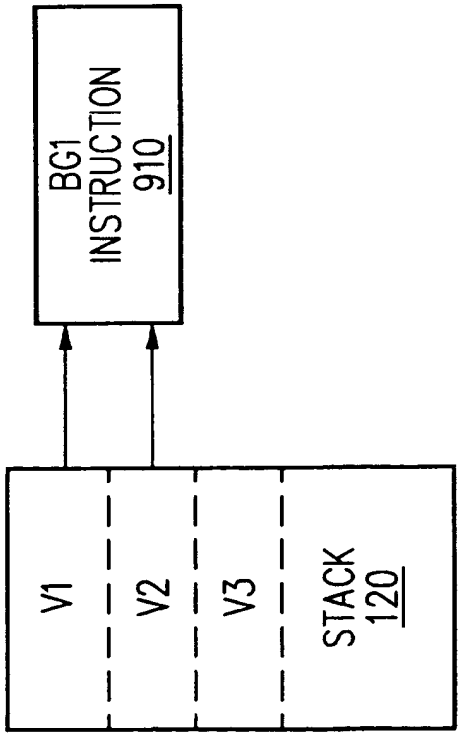

FIG. 9a

	Document ID	U	Title	Current OR
75	US 6502181 B1	☒	Method and apparatus for an enhanced processor	712/32
76	US 6484251 B1	☒	Updating condition status register based on instruction specific modification information in set/clear pair upon instruction commit in out-of-order processor	712/23
77	US 6438621 B1	☒	In-memory modification of computer programs	719/331
78	US 6401193 B1	☒	Dynamic data prefetching based on program counter and addressing mode	712/207
79	US 6393553 B1	☒	Acknowledgement mechanism for just-in-time delivery of load data	712/217
80	US 6370632 B1	☒	Method and apparatus that enforces a regional memory model in hierarchical memory systems	711/205
81	US 6360318 B1	☒	Configurable branch prediction for a processor performing speculative execution	712/240
82	US 6338136 B1	☒	Pairing of load-ALU-store with conditional branch	712/221
83	US 6320110 B1	☒	Music game device with automatic setting, method for controlling the same, and storage medium therefor	84/600
84	US 6317821 B1	☒	Virtual single-cycle execution in pipelined processors	712/200
85	US 6314511 B1	☒	Mechanism for freeing registers on processors that perform dynamic out-of-order execution of instructions using renaming registers	712/217
86	US 6282663 B1	☒	Method and apparatus for performing power management by suppressing the speculative execution of instructions within a pipelined microprocessor	713/320
87	US 6263926 B1	☒	Configurable branch prediction for a processor performing speculative execution	712/240
88	US 6263930 B1	☒	High-performance, superscalar-based computer system with out-of-order instruction execution and concurrent results distribution	712/23
89	US 6247107 B1	☒	Branch selectors associated with byte ranges within an instruction cache for rapidly identifying branch predictions	712/239
90	US 6226719 B1	☒	High-performance, superscalar-based computer system with out-of-order instruction execution	712/41
91	US 6201731 B1	☒	Method and apparatus for TLB memory ordering	711/210
92	US 6256720 B1	☒	High performance, superscalar-based computer system with out-of-order instruction execution	712/23
93	US 6253316 B1	☒	Three state branch history using one bit in a branch prediction mechanism	712/239
94	US 6247123 B1	☒	Branch prediction mechanism employing branch selectors to select a branch prediction	712/239
95	US 6247097 B1	☒	Aligned instruction cache handling of instruction fetches across multiple predicted branch instructions	711/125
96	US 6237076 B1	☒	Method for register renaming by copying a 32 bits instruction directly or indirectly to a 64 bits instruction	712/23
97	US 6230260 B1	☒	Circuit arrangement and method of speculative instruction execution utilizing instruction history caching	712/239

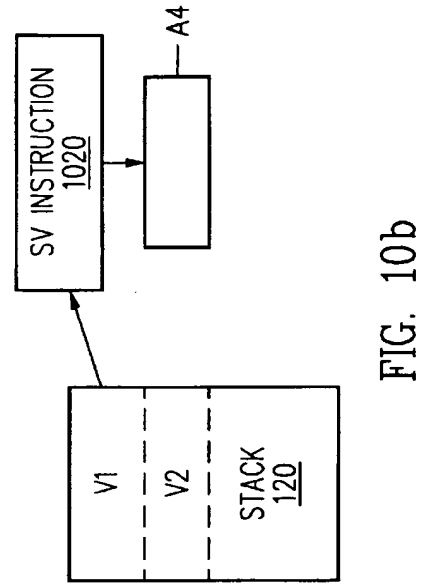

FIG. 10b

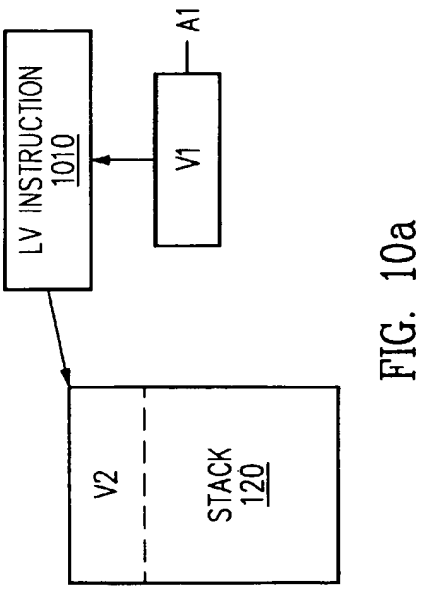

FIG. 10a

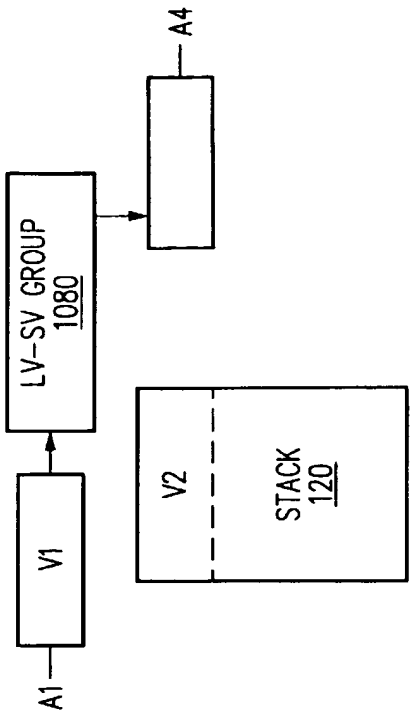

FIG. 10d

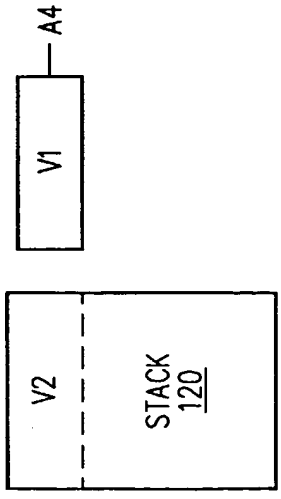

FIG. 10c

	Document ID	U	Title	Current OR
98	US 6216676 B1	☒	Internal combustion engine system	123/568.21
99	US 6216143 B1	☒	Apparatus and method for generating animated color coded software traces	715/526
100	US 6189091 B1	☒	Apparatus and method for speculatively updating global history and restoring same on branch misprediction detection	712/240
101	US 6178441 B1	☒	Method and system in a computer network for the reliable and consistent ordering of client requests	709/203
102	US 6178438 B1	☒	Service management system for an advanced intelligent network	709/200
103	US 6148292 A	☒	Method for statistics mode reloading and for statistical acquisition according to statistics classes in the storing of a dataset	705/30
104	US 6141748 A	☒	Branch selectors associated with byte ranges within an instruction cache for rapidly identifying branch predictions	712/239
105	US 6137588 A	☒	Digital reproduction control method and apparatus having autonomous and command control modes	358/1.15
106	US 6128723 A	☒	High-performance, superscalar-based computer system with out-of-order instruction execution	712/23
107	US 6119222 A	☒	Combined branch prediction and cache prefetch in a microprocessor	712/238
108	US 6108777 A	☒	Configurable branch prediction for a processor performing speculative execution	712/240
109	US 6108774 A	☒	Branch prediction with added selector bits to increase branch prediction capacity and flexibility with minimal added bits	712/240
110	US 6101594 A	☒	High-performance, superscalar-based computer system with out-of-order instruction execution	712/41
111	US 6092181 A	☒	High-performance, superscalar-based computer system with out-of-order instruction execution	712/206
112	US 6061823 A	☒	Error correcting/decoding apparatus and error correcting/decoding method	714/758
113	US 6055626 A	☒	Method and circuit for delayed branch control and method and circuit for conditional-flag rewriting control	712/216
114	US 6055624 A	☒	Millicode flags with specialized update and branch instructions	712/208
115	US 6047369 A	☒	Flag renaming and flag masks within register alias table	712/217
116	US 6038654 A	☒	High performance, superscalar-based computer system with out-of-order instruction execution	712/23
117	US 6038653 A	☒	High-performance superscalar-based computer system with out-of-order instruction execution and concurrent results distribution	712/23
118	US 6031631 A	☒	Data processing system and method using virtual storage system	710/260
119	US 6029240 A	☒	Method for processing instructions for parallel execution including storing instruction sequences along with compounding information in cache	712/23
120	US 6029146 A	☒	Method and apparatus for trading securities electronically	705/35

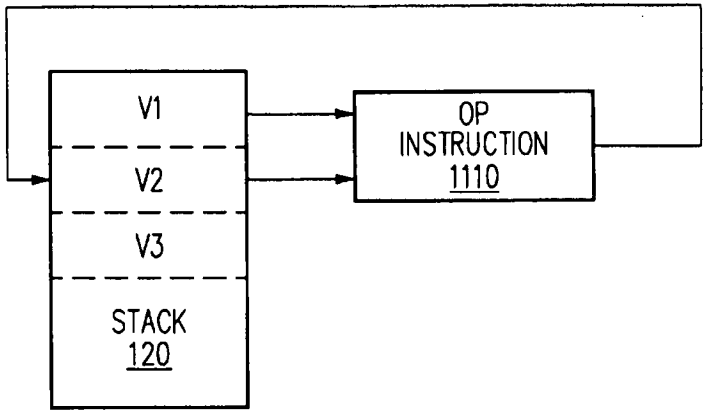

FIG. 11a

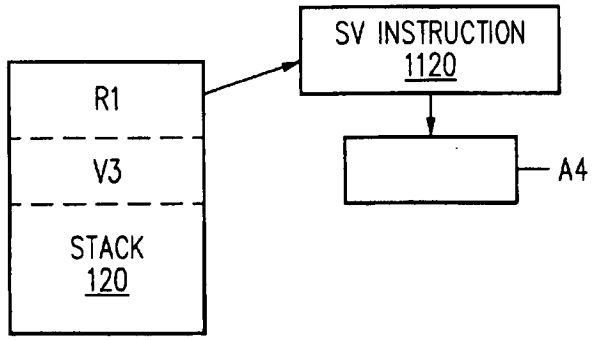

FIG. 11b

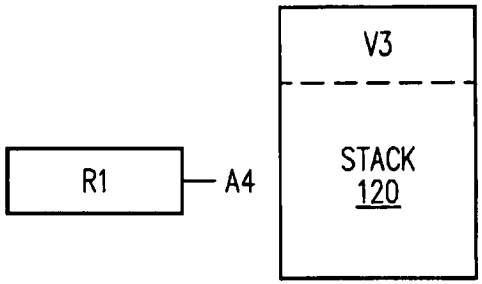

FIG. 11c

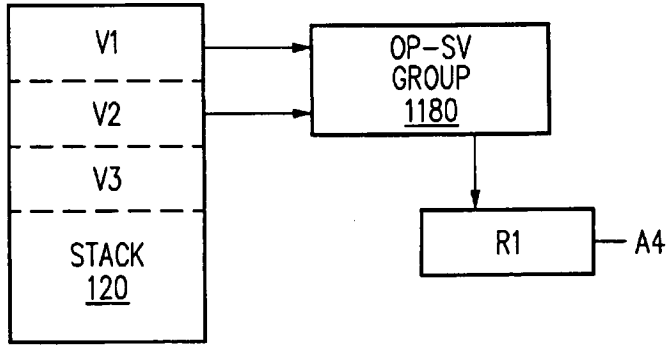

FIG. 11d

	Document ID	U	Title	Current OR
121	US 6021489 A	☒	Apparatus and method for sharing a branch prediction unit in a microprocessor implementing a two instruction set architecture	712/239
122	US 6003128 A	☒	Number of pipeline stages and loop length related counter differential based end-loop prediction	712/241
123	US 6002850 A	☒	Operation sequence user adaptive system and method	719/315
124	US 5996071 A	☒	Detecting self-modifying code in a pipelined processor with branch processing by comparing latched store address to subsequent target address	712/238
125	US 5996069 A	☒	Method and circuit for delayed branch control and method and circuit for conditional-flag rewriting control	712/234
126	US 5995749 A	☒	Branch prediction mechanism employing branch selectors to select a branch prediction	712/239
127	US 5987603 A	☒	Apparatus and method for reversing bits using a shifter	712/300
128	US 5978906 A	☒	Branch selectors associated with byte ranges within an instruction cache for rapidly identifying branch predictions	712/239
129	US 5974542 A	☒	Branch prediction unit which approximates a larger number of branch predictions using a smaller number of branch predictions and an alternate target indication	712/239
130	US 5974535 A	☒	Method and system in data processing system of permitting concurrent processing of instructions of a particular type	712/215
131	US 5966544 A	☒	Data speculatable processor having reply architecture	712/32
132	US 5961638 A	☒	Branch prediction mechanism employing branch selectors to select a branch prediction	712/239
133	US 5961629 A	☒	High performance, superscalar-based computer system with out-of-order instruction execution	712/23
134	US 5956495 A	☒	Method and system for processing branch instructions during emulation in a data processing system	703/26
135	US 5954816 A	☒	Branch selector prediction	712/239
136	US 5953520 A	☒	Address translation buffer for data processing system emulation mode	703/26
137	US 5951678 A	☒	Method and apparatus for controlling conditional branch execution in a data processor	712/237
138	US 5933860 A	☒	Multiprobe instruction cache with instruction-based probe hint generation and training whereby the cache bank or way to be accessed next is predicted	711/213
139	US 5930521 A	☒	Reorder buffer architecture for accessing partial word operands	712/23
140	US 5923862 A	☒	Processor that decodes a multi-cycle instruction into single-cycle micro-instructions and schedules execution of the micro-instructions	712/208
141	US 5920710 A	☒	Apparatus and method for modifying status bits in a reorder buffer with a large speculative state	712/216
142	US 5918030 A	☒	Device for executing a program of instructions	712/206
143	US 5903750 A	☒	Dynamic branch prediction for branch instructions with multiple targets	712/236

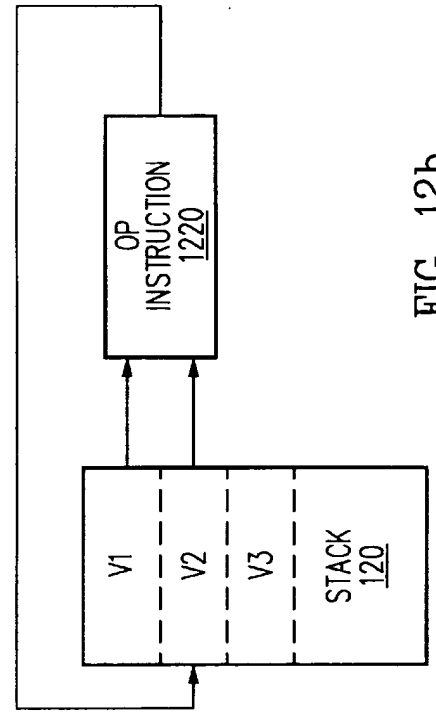

FIG. 12b

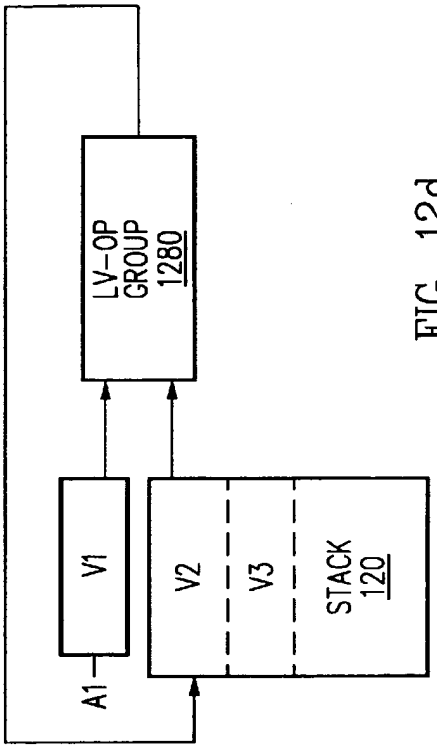

FIG. 12d

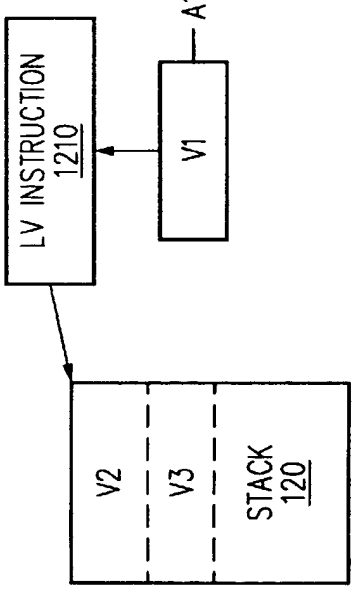

FIG. 12a

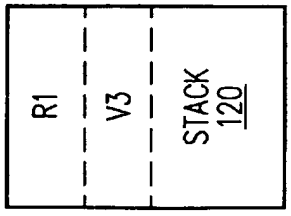

FIG. 12c

	Document ID	U	Title	Current OR
144	US 5899203 A	☒	Interactive respiratory regulator	128/204.23
145	US 5893165 A	☒	System and method for parallel execution of memory transactions using multiple memory models, including SSO, TSO, PSO and RMO	711/158
146	US 5892963 A	☒	System and method for assigning tags to instructions to control instruction execution	712/23
147	US 5884061 A	☒	Apparatus to perform source operand dependency analysis perform register renaming and provide rapid pipeline recovery for a microprocessor capable of issuing and executing multiple instructions out-of-order in a single processor cycle	712/217
148	US 5875326 A	☒	Data processing system and method for completing out-of-order instructions	712/244
149	US 5872965 A	☒	System and method for performing multiway branches using a visual instruction set	712/236
150	US 5870579 A	☒	Reorder buffer including a circuit for selecting a designated mask corresponding to an instruction that results in an exception	712/217
151	US 5870575 A	☒	Indirect unconditional branches in data processing system emulation mode	712/209
152	US 5867663 A	☒	Graphical user interface for defining and invoking user-customized tool shelf execution sequence	345/840
153	US 5850521 A	☒	Apparatus and method for interprocessor communication	709/208
154	US 5848269 A	☒	Branch predicting mechanism for enhancing accuracy in branch prediction by reference to data	712/239
155	US 5832292 A	☒	High-performance superscalar-based computer system with out-of-order instruction execution and concurrent results distribution	712/23
156	US 5828860 A	☒	Data processing device equipped with cache memory and a storage unit for storing data between a main storage or CPU cache memory	712/207
157	US 5822778 A	☒	Microprocessor and method of using a segment override prefix instruction field to expand the register file	711/208
158	US 5819080 A	☒	Microprocessor using an instruction field to specify condition flags for use with branch instructions and a computer system employing the microprocessor	712/239
159	US 5815699 A	☒	Configurable branch prediction for a processor performing speculative execution	712/239
160	US 5809294 A	☒	Parallel processing unit which processes branch instructions without decreased performance when a branch is taken	712/233
161	US 5805878 A	☒	Method and apparatus for generating branch predictions for multiple branch instructions indexed by a single instruction pointer	712/239
162	US 5802412 A	☒	Miniature pan/tilt tracking mount	396/427
163	US 5799289 A	☒	Order management system and method considering budget limit	705/400
164	US 5794031 A	☒	Distributed processing system for system booting and shutdown in distributed processing environment	713/2
165	US 5794028 A	☒	Shared branch prediction structure	712/240

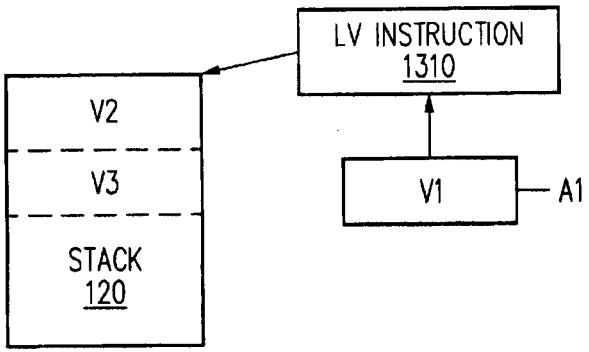

FIG. 13a

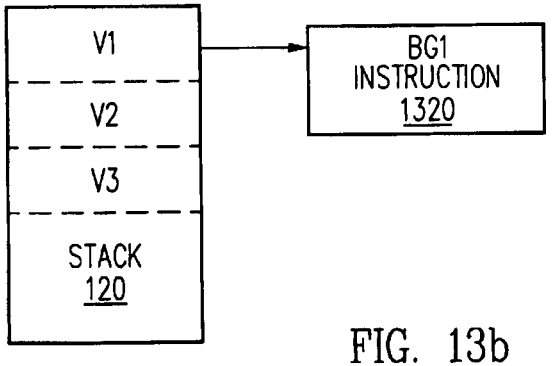

FIG. 13b

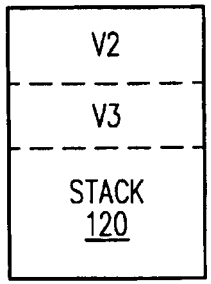

FIG. 13c

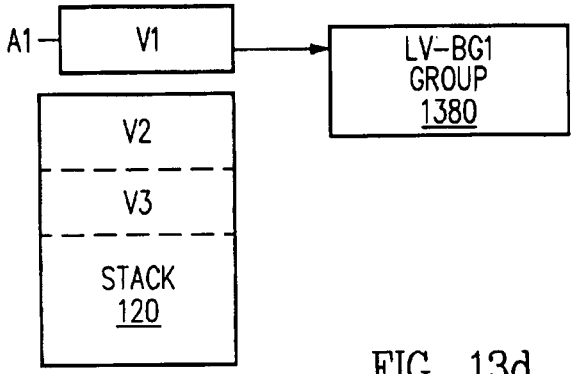

FIG. 13d

	Document ID	U	Title	Current OR
166	US 5784604 A	☒	Method and system for reduced run-time delay during conditional branch execution in pipelined processor systems utilizing selectively delayed sequential instruction purging	712/238
167	US 5784603 A	☒	Fast handling of branch delay slots on mispredicted branches	712/234
168	US 5774712 A	☒	Instruction dispatch unit and method for mapping a sending order of operations to a receiving order	712/245
169	US 5774685 A	☒	Method and apparatus for biasing cache LRU for prefetched instructions/data based upon evaluation of speculative conditions	712/205
170	US 5768574 A	☒	Microprocessor using an instruction field to expand the condition flags and a computer system employing the microprocessor	712/226
171	US 5765221 A	☒	Method and system of addressing which minimize memory utilized to store logical addresses by storing high order bits within a register	711/220
172	US 5761490 A	☒	Changing the meaning of a pre-decode bit in a cache memory depending on branch prediction mode	712/239
173	US 5761474 A	☒	Operand dependency tracking system and method for a processor that executes instructions out of order	712/217
174	US 5761469 A	☒	Method and apparatus for optimizing signed and unsigned load processing in a pipelined processor	712/210
175	US 5737576 A	☒	Method and system for efficient instruction execution in a data processing system having multiple prefetch units	711/169
176	US 5735202 A	☒	Print sorting method for image forming apparatus with sorter and control system for executing such print sorting method	101/2
177	US 5734904 A	☒	Method and system for calling one of a set of routines designed for direct invocation by programs of a second type when invoked by a program of the first type	719/331
178	US 5732254 A	☒	Pipeline system branch history table storing branch instruction addresses and target addresses with inhibit bits	712/240
179	US 5730145 A	☒	Interactive respiratory regulator	600/595
180	US 5701448 A	☒	Detecting segment limit violations for branch target when the branch unit does not supply the linear address	712/233
181	US 5699506 A	☒	Method and apparatus for fault testing a pipelined processor	714/37
182	US 5692168 A	☒	Prefetch buffer using flow control bit to identify changes of flow within the code stream	712/237
183	US 5689720 A	☒	High-performance superscalar-based computer system with out-of-order instruction execution	712/23
184	US 5680578 A	☒	Microprocessor using an instruction field to specify expanded functionality and a computer system employing same	711/154
185	US 5666506 A	☒	Apparatus to dynamically control the out-of-order execution of load/store instructions in a processor capable of dispatchng, issuing and executing multiple instructions in a single processor cycle	712/216
186	US 5655096 A	☒	Method and apparatus for dynamic scheduling of instructions to ensure sequentially coherent data in a processor employing out-of-order execution	712/200
187	US 5652858 A	☒	Method for prefetching pointer-type data structure and information processing apparatus therefor	711/137

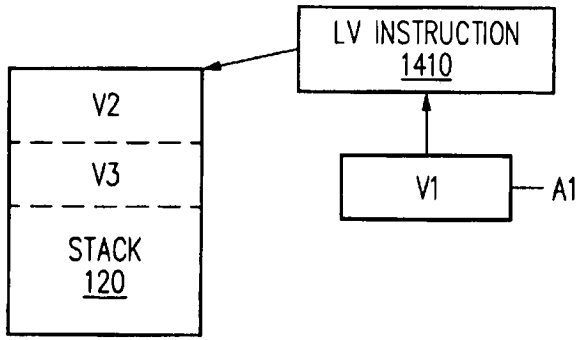

FIG. 14a

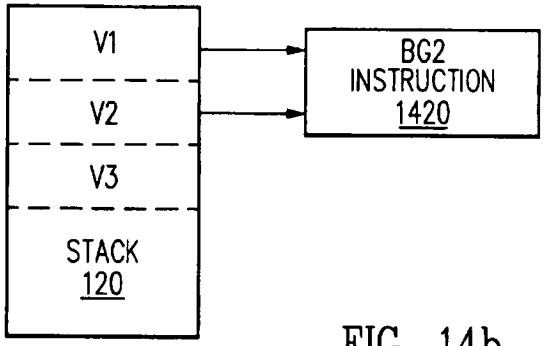

FIG. 14b

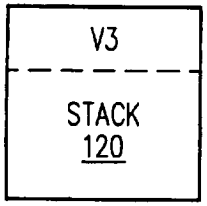

FIG. 14c

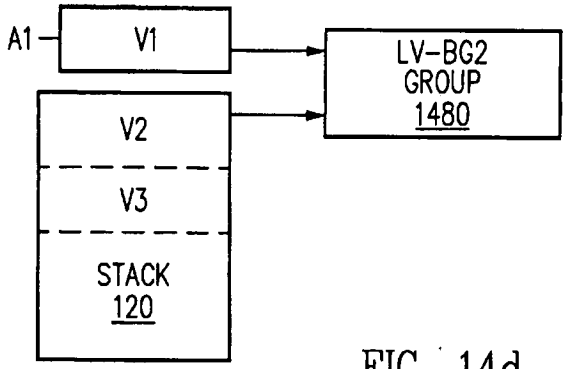

FIG. 14d

	Document ID	U	Title	Current OR
188	US 5650341 A	☒	Process for fabricating CMOS Device	438/217
189	US 5644759 A	☒	Apparatus and method for processing a jump instruction preceded by a skip instruction	712/240
190	US 5627984 A	☒	Apparatus and method for entry allocation for a buffer resource utilizing an internal two cycle pipeline	712/200
191	US 5625835 A	☒	Method and apparatus for reordering memory operations in a superscalar or very long instruction word processor	712/23
192	US 5625789 A	☒	Apparatus for source operand dependendency analyses register renaming and rapid pipeline recovery in a microprocessor that issues and executes multiple instructions out-of-order in a single cycle	712/217
193	US 5615350 A	☒	Apparatus to dynamically control the out-of-order execution of load-store instructions in a processor capable of dispatching, issuing and executing multiple instructions in a single processor cycle	712/218
194	US 5608886 A	☒	Block-based branch prediction using a target finder array storing target sub-addresses	712/239
195	US 5604912 A	☒	System and method for assigning tags to instructions to control instruction execution	712/23
196	US 5598546 A	☒	Dual-architecture super-scalar pipeline	712/209
197	US 5592684 A	☒	Store queue including a byte order tracking mechanism for maintaining data coherency	710/52
198	US 5592634 A	☒	Zero-cycle multi-state branch cache prediction data processing system and method thereof	712/239
199	US 5577259 A	☒	Instruction processor control system using separate hardware and microcode control signals to control the pipelined execution of multiple classes of machine instructions	712/41
200	US 5560032 A	☐	High-performance, superscalar-based computer system with out-of-order instruction execution and concurrent results distribution	712/23

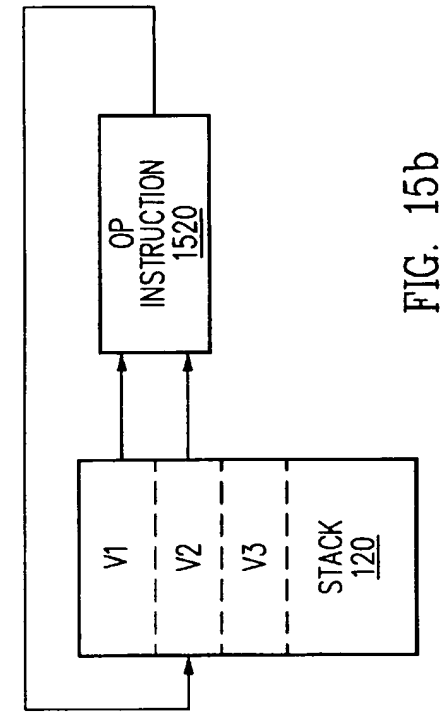

FIG. 15b

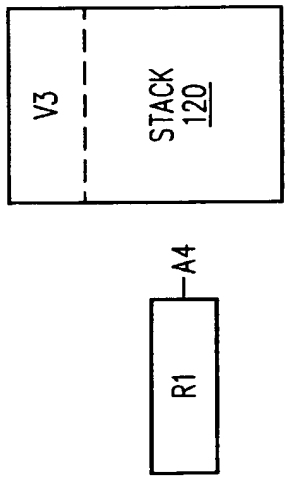

FIG. 15d

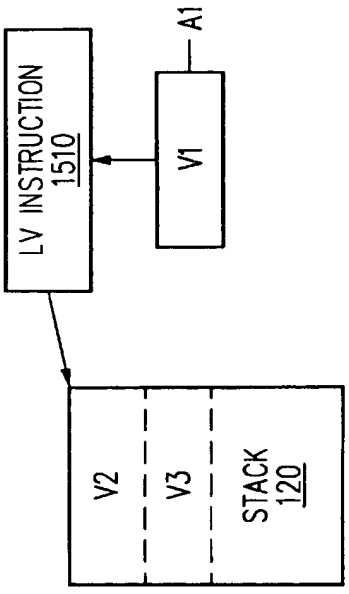

FIG. 15a

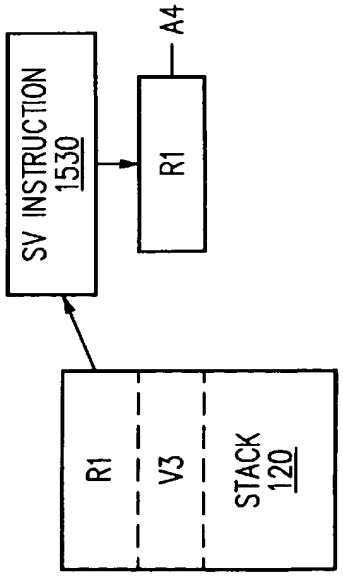

FIG. 15c

	Document ID	U	Title	Current OR
1	US 20040030838 A1	☐	Instruction cache way prediction for jump targets	711/137
2	US 20040024999 A1	☒	Micro-sequence execution in a processor	712/245
3	US 20040006957 A1	☒	PROGRAMMABLE FUNCTION CONTROL FOR COMBINE	56/10.2 G
4	US 20040006728 A1	☒	Method and device for simultaneous testing of a plurality of integrated circuits	714/724
5	US 20030233559 A1	☒	Data processing apparatus and data processing method	713/189
6	US 20030210246 A1	☒	Network management card for use in a system for screen image capturing	345/520
7	US 20030187905 A1	☒	Scheduling tasks quickly in a sequential order	718/100
8	US 20030144012 A1	☒	Information providing apparatus, computer program product and information providing method	455/456.1
9	US 20030140253 A1	☒	Method of and apparatus for detecting creation of set user identification (setuid) files, and computer program for enabling such detection	713/201
10	US 20030120790 A1	☒	Processor with multiple-pass non-sequential packet classification feature	709/230
11	US 20030119497 A1	☒	Test system for remotely testing switches within a telecommunications network	455/424
12	US 20030109975 A1	☒	Agricultural vehicle	701/50
13	US 20030093736 A1	☒	System and method enabling hierarchical execution of a test executive subsequence	714/738
14	US 20030079109 A1	☒	Methods and apparatus for dynamic very long instruction word sub-instruction selection for execution time parallelism in an indirect very long instruction word processor	712/24
15	US 20030070051 A1	☒	Methods and apparatus for utilizing flash burst mode to improve processor performance	711/167
16	US 20030026249 A1	☒	Inter-nodal data transfer system and data transfer apparatus	370/360
17	US 20030023847 A1	☒	DATA PROCESSING SYSTEM, RECORDING DEVICE, DATA PROCESSING METHOD AND PROGRAM PROVIDING MEDIUM	713/169

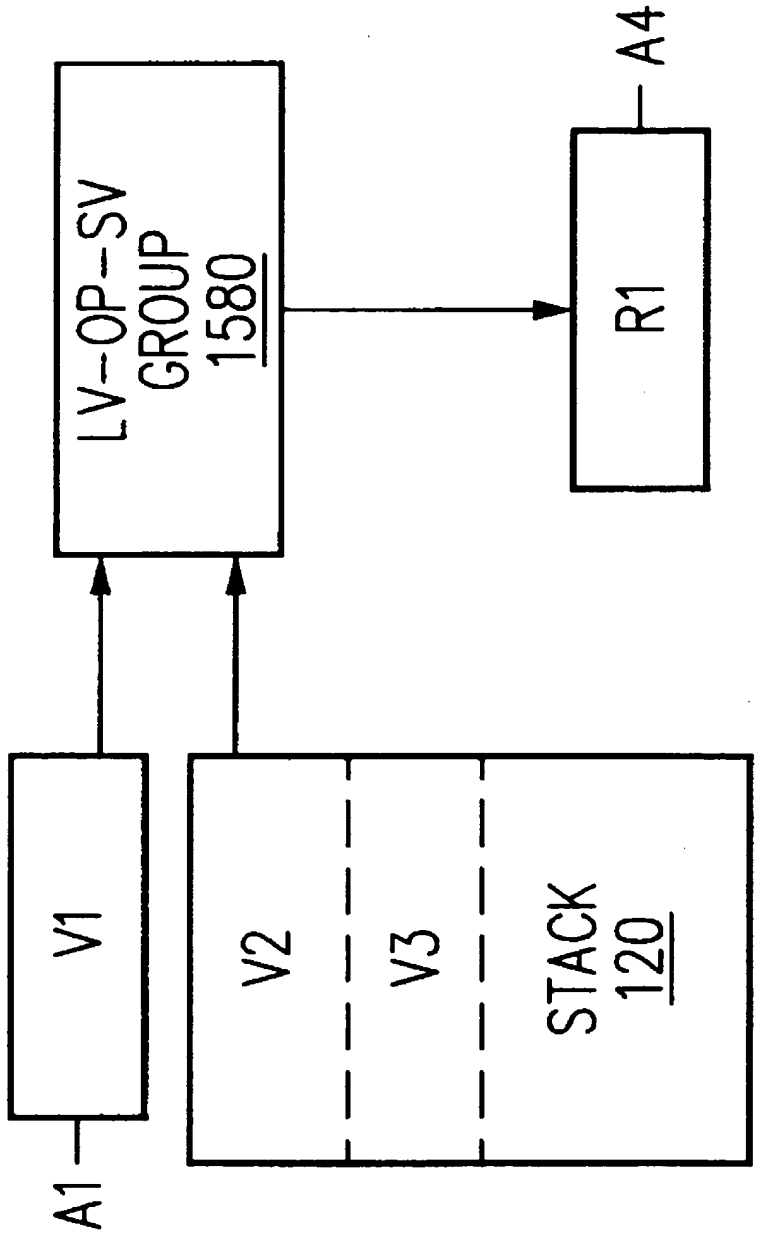

FIG. 15e

	Document ID	U	Title	Current OR
18	US 20030 00931 3 A1	☒	Real-time alert mechanism for monitoring and controlling field assets via wireless and internet technologies	702/188
19	US 20030 00253 7 A1	☒	Method and apparatus for controlling the timing of a communication device	370/503
20	US 20020 15704 2 A1	☒	Algorithmically programmable memory tester with breakpoint trigger, error jamming and 'scope mode that memorizes target sequences	714/45
21	US 20020 11157 0 A1	☒	MICROCONTROLLER BASED MASSAGE SYSTEM	601/15
22	US 20020 10803 0 A1	☒	Method and system for performing permutations using permutation instructions based on modified omega and flip stages	712/300
23	US 20020 07832 0 A1	☒	Methods and apparatus for instruction addressing in indirect VLIW processors	712/24
24	US 20020 03122 0 A1	☒	Method and system for performing permutations using permutation instructions based on butterfly networks	380/37
25	US 20020 02865 9 A1	☒	Test system for remotely testing swithches within a telecommunications network	455/67.11
26	US 20010 02917 9 A1	☒	Test system for remotely testing switches within a telecommunications network	455/423
27	US 66815 51 B1	☒	Programmable function control for combine	56/10.2 G
28	US 66788 15 B1	☒	Apparatus and method for reducing power consumption due to cache and TLB accesses in a processor front-end	711/205
29	US 66657 92 B1	☒	Interface to a memory system for a processor having a replay system	712/219
30	US 66648 33 B1	☒	Dual-edge function clock generator and method of deriving clocking signals for executing reduced instruction sequences in a re-programmable I/O interface	327/170
31	US 66548 78 B1	☒	Register bit scanning	712/234
32	US 65811 52 B2	☒	Methods and apparatus for instruction addressing in indirect VLIW processors	712/24
33	US 65781 33 B1	☒	MIMD array of single bit processors for processing logic equations in strict sequential order	712/21
34	US 65670 84 B1	☒	Lighting effect computation circuit and method therefore	345/426
35	US 65562 08 B1	☒	Network management card for use in a system for screen image capturing	345/520
36	US 65427 38 B2	☒	Test system for remotely testing switches within a telecommunications network	455/424
37	US 65421 99 B1	☒	Cadence editing	348/459

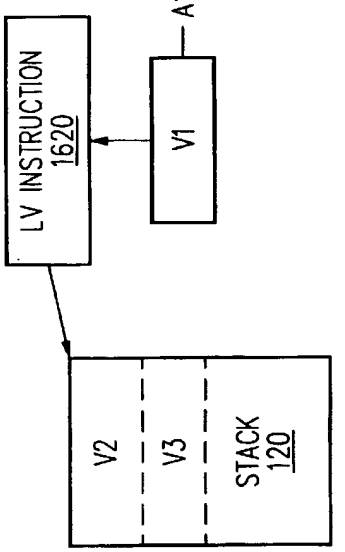

FIG. 16b

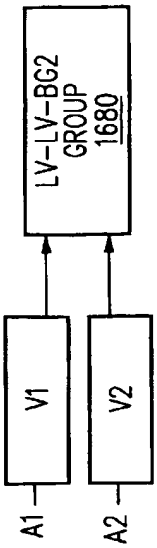

FIG. 16e

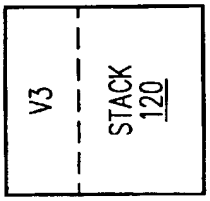

FIG. 16d

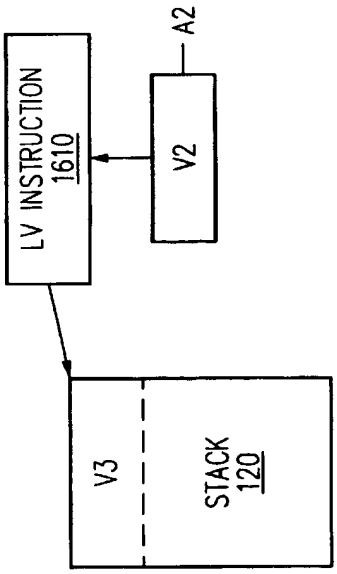

FIG. 16a

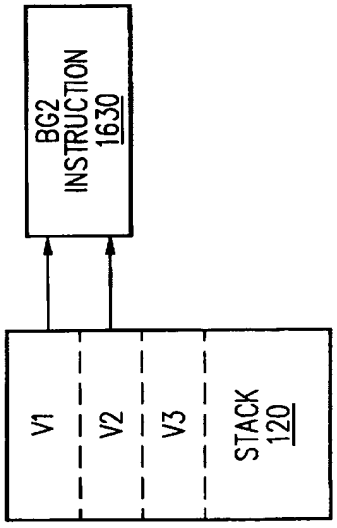

FIG. 16c

	Document ID	U	Title	Current OR
38	US 6538998 B1	☒	Rolling out high bandwidth connection services in geographical areas covering several central offices	370/241
39	US 6536034 B1	☒	Method for modifying code sequences and related device	717/110
40	US 6507921 B1	☒	Trace fifo management	714/45
41	US 6469704 B1	☒	System and method for combined execution of graphics primitive data sets	345/553
42	US 6467036 B1	☒	Methods and apparatus for dynamic very long instruction word sub-instruction selection for execution time parallelism in an indirect very long instruction word processor	712/24
43	US 6457152 B1	☒	Device and method for testing a device through resolution of data into atomic operations	714/738
44	US 6453412 B1	☒	Method and apparatus for reissuing paired MMX instructions singly during exception handling	712/244
45	US 6418336 B1	☒	MR tomography apparatus and method for suppressing stimulation of an examination subject	600/410
46	US 6370146 B1	☒	Method and apparatus for non-disruptive addition of a new node to an inter-nodal network	370/400
47	US 6362825 B1	☒	Real-time combination of adjacent identical primitive data sets in a graphics call sequence	345/522
48	US 6356994 B1	☒	Methods and apparatus for instruction addressing in indirect VLIW processors	712/24
49	US 6275982 B1	☒	Method and device enabling a fixed program to be developed	717/168
50	US 6275920 B1	☒	Mesh connected computed	712/14
51	US 6253306 B1	☒	Prefetch instruction mechanism for processor	712/207
52	US 6253287 B1	☒	Using three-dimensional storage to make variable-length instructions appear uniform in two dimensions	711/125
53	US 6243848 B1	☒	Process for analyzing complex structures and system for implementing a process of this type	716/1
54	US 6240508 B1	☒	Decode and execution synchronized pipeline processing using decode generated memory read queue with stop entry to allow execution generated memory read	712/219
55	US 6230278 B1	☒	Microprocessor with functional units that can be selectively coupled	713/324
56	US 6230006 B1	☒	Test system for remotely testing switches within a telecommunications network	455/424
57	US 6212628 B1	☒	Mesh connected computer	712/226
58	US 6173389 B1	☒	Methods and apparatus for dynamic very long instruction word sub-instruction selection for execution time parallelism in an indirect very long instruction word processor	712/24
59	US 6170038 B1	☒	Trace based instruction caching	711/125
60	US 6167330 A	☒	Dynamic power management of systems	700/295

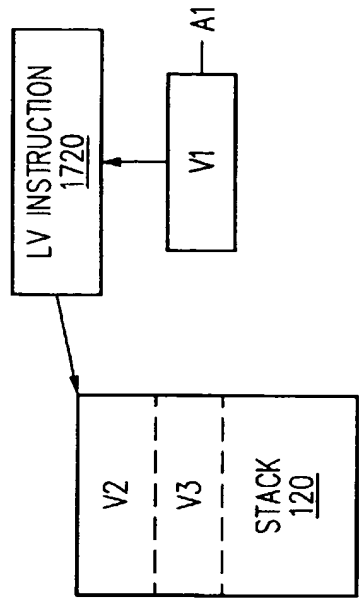

FIG. 17b

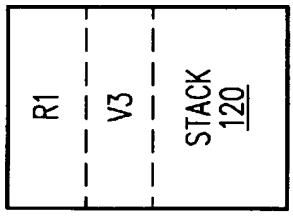

FIG. 17d

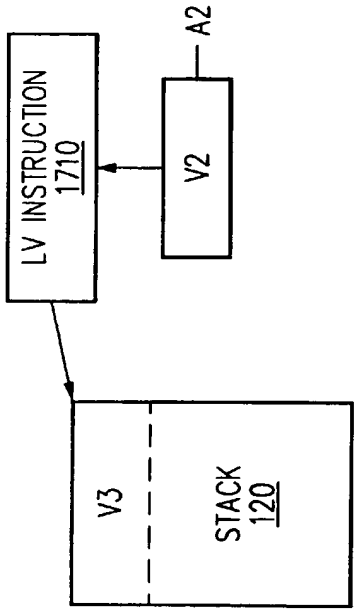

FIG. 17a

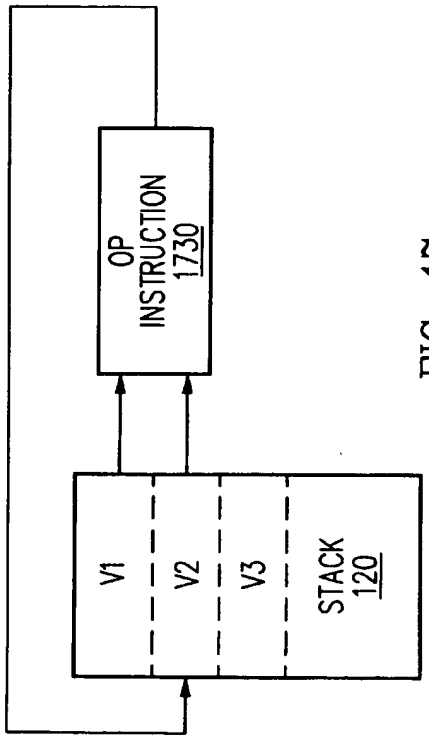

FIG. 17c

	Document ID	U	Title	Current OR
61	US 6157964 A	☒	Method for specifying concurrent execution of a string of I/O command blocks in a chain structure	710/5
62	US 6151705 A	☒	Efficient use of the base register auto-increment feature of memory access instructions	717/153
63	US 6144321 A	☒	Microprocessor dedicated to processing of bit streams in a system of compression/decompression of animated images	341/67
64	US 6133938 A	☒	Descriptor mechanism for assuring indivisible execution of AV/C operations	725/80
65	US 6133859 A	☒	Microprocessor dedicated to processing of bit streams in a system of compression/decompression of animated images	341/67
66	US 6123174 A	☒	Apparatus and method for automatically performing fluid changes	184/1.5
67	US 6122320 A	☒	Circuit for motion estimation in digitized video sequence encoders	375/240
68	US 6091930 A	☒	Customizable interactive textbook	434/362
69	US 6085314 A	☒	Central processing unit including APX and DSP cores and including selectable APX and DSP execution modes	712/213
70	US 6079008 A	☒	Multiple thread multiple data predictive coded parallel processing system and method	712/11
71	US 6076144 A	☒	Method and apparatus for identifying potential entry points into trace segments	711/125
72	US 6065040 A	☒	Computer system having agent retracting method and agent returning method	709/202
73	US 6039702 A	☒	Microcontroller based massage system	601/15
74	US 6032247 A	☒	Central processing unit including APX and DSP cores which receives and processes APX and DSP instructions	712/35
75	US 6018786 A	☒	Trace based instruction caching	711/4
76	US 6016544 A	☒	Apparatus and method for tracking changes in address size and for different size retranslate second instruction with an indicator from address size	712/234
77	US 6011908 A	☒	Gated store buffer for an advanced microprocessor	714/19
78	US 5983324 A	☒	Data prefetch control method for main storage cache for protecting prefetched data from replacement before utilization thereof	711/137
79	US 5962839 A	☒	Apparatus programmable to perform a user defined sequence of actions	235/472.01
80	US 5958045 A	☒	Start of access instruction configured to indicate an access mode for fetching memory operands in a microprocessor	712/229
81	US 5941980 A	☒	Apparatus and method for parallel decoding of variable-length instructions in a superscalar pipelined data processing system	712/204
82	US 5933850 A	☒	Instruction unit having a partitioned cache	711/125
83	US 5926644 A	☒	Instruction formats/instruction encoding	712/22

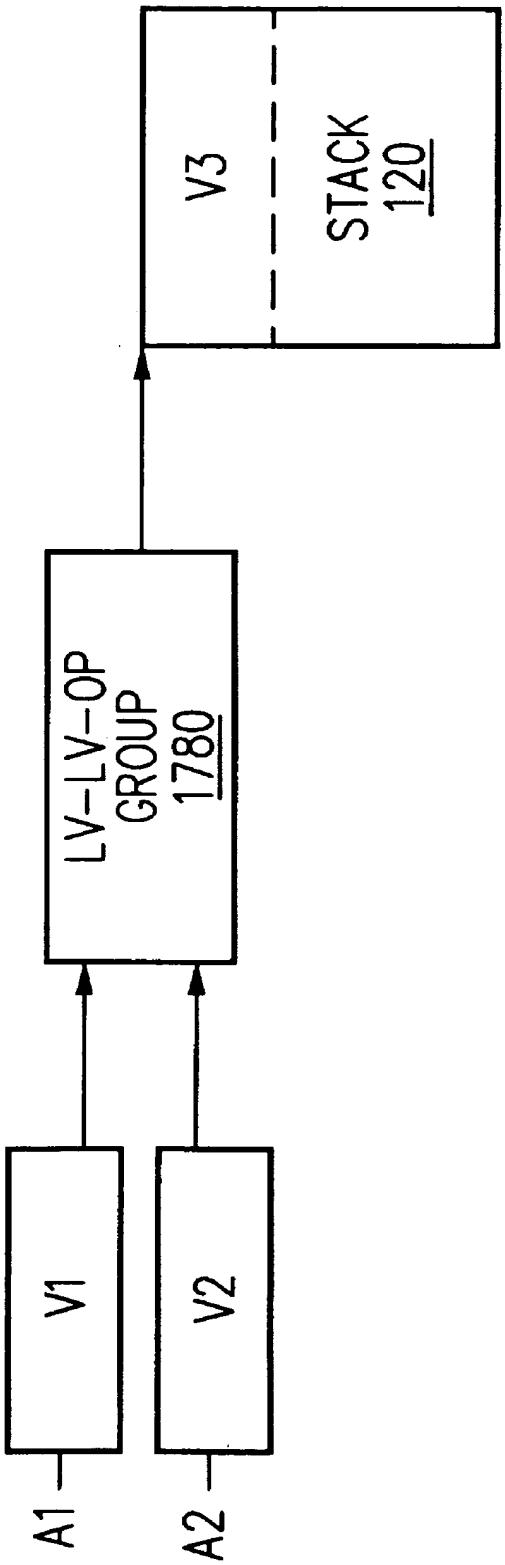

FIG. 17e

	Document ID	U	Title	Current OR
84	US 5923896 A	☒	Method for sequencing execution of I/O command blocks in a chain structure by setting hold-off flags and configuring a counter in each I/O command block	710/5
85	US 5922065 A	☒	Processor utilizing a template field for encoding instruction sequences in a wide-word format	712/24
86	US 5905883 A	☒	Verification system for circuit simulator	703/17
87	US 5898120 A	☒	Auto-play apparatus for arpeggio tones	84/638
88	US 5892969 A	☒	Method for concurrently executing a configured string of concurrent I/O command blocks within a chain to perform a raid 5 I/O operation	710/5
89	US 5892906 A	☒	Apparatus and method for preventing theft of computer devices	713/202
90	US 5881263 A	☒	Non-instruction base register addressing in a data processing apparatus	712/217
91	US 5881259 A	☒	Input operand size and hi/low word selection control in data processing systems	712/210
92	US 5857088 A	☒	System for configuring memory space for storing single decoder table, reconfiguring same space for storing plurality of decoder tables, and selecting one configuration based on encoding scheme	712/210
93	US 5852729 A	☒	Code segment replacement apparatus and real time signal processor using same	712/225
94	US 5850567 A	☒	Method for specifying concurrent execution of a string of I/O command blocks in a chain structure	710/5
95	US 5848026 A	☒	Integrated circuit with flag register for block selection of nonvolatile cells for bulk operations	365/238.5
96	US 5836989 A	☒	Method and apparatus for controlling an implanted medical device in a time-dependent manner	607/27
97	US 5809320 A	☒	High-performance multi-processor having floating point unit	712/34
98	US 5802338 A	☒	Method of self-parallelizing and self-parallelizing multiprocessor using the method	712/217
99	US 5797034 A	☒	Method for specifying execution of only one of a pair of I/O command blocks in a chain structure	710/24
100	US 5794063 A	☒	Instruction decoder including emulation using indirect specifiers	712/23
101	US 5793761 A	☒	Communication switching system having a user facility system and a base switching system	370/377
102	US 5768621 A	☒	Chain manager for use in executing a chain of I/O command blocks	710/24
103	US 5761477 A	☒	Methods for safe and efficient implementations of virtual machines	718/1
104	US 5758187 A	☒	Method for enhancing performance of a RAID 1 read operation using a pair of I/O command blocks in a chain structure	710/24
105	US 5742822 A	☒	Multithreaded processor which dynamically discriminates a parallel execution and a sequential execution of threads	718/102

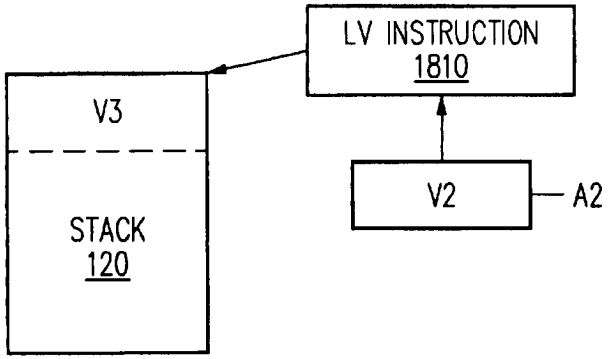

FIG. 18a

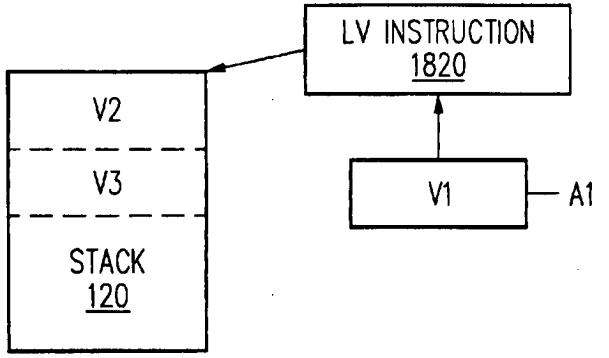

FIG. 18b

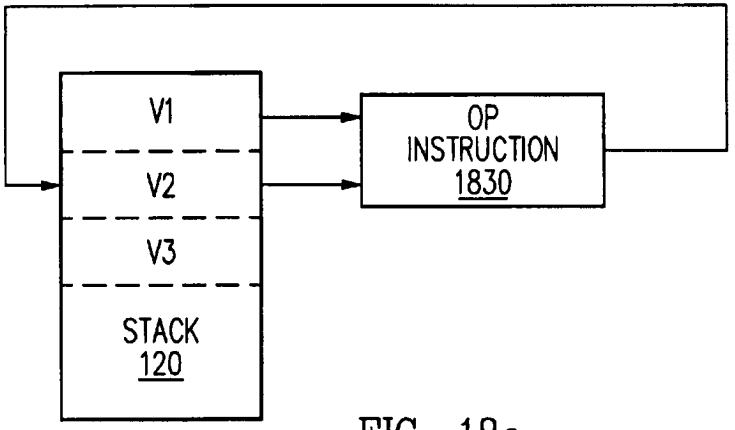

FIG. 18c

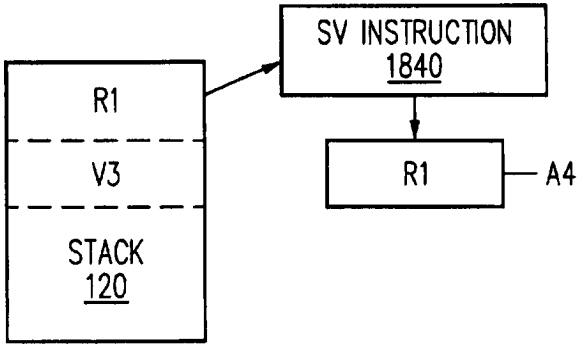

FIG. 18d

	Document ID	U	Title	Current OR
106	US 5732234 A	☒	System for obtaining parallel execution of existing instructions in a particulr data processing configuration by compounding rules based on instruction categories	712/200
107	US 5701450 A	☒	System including ATA sequencer microprocessor which executes sequencer instructions to handle plurality of real-time events allowing to perform all operations without local microprocessor intervention	712/245
108	US 5701435 A	☒	Instruction cache system for implementing programs having non-sequential instructions and method of implementing same	711/159
109	US 5685396 A	☒	Apparatus for automatically performing engine fluid changes	184/1.5
110	US 5684972 A	☒	Programmable servo burst sequencer for a disk drive	711/4
111	US 5678048 A	☒	Interrupt vector method and apparatus for loading a slot memory address counter	710/266
112	US 5640583 A	☒	Programmable servo burst decoder	713/600
113	US 5640538 A	☒	Programmable timing mark sequencer for a disk drive	703/23
114	US 5640509 A	☒	Programmable built-in self-test function for an integrated circuit	714/42
115	US 5631974 A	☒	Image processing	382/132
116	US 5612834 A	☒	Servo burst controller for a magnetic disk	360/75
117	US 5600807 A	☒	Programmable controller capable of updating a user program during operation by switching between user program memories	711/211
118	US 5596737 A	☒	Sequencer map for a hard disk controller combining data and next-addres fields	711/213
119	US 5592348 A	☒	Method and structure for locating and skipping over servo bursts on a magnetic disk	360/77.08
120	US 5572666 A	☒	System and method for generating pseudo-random instructions for design verification	714/32
121	US 5564023 A	☒	Method for accessing a sequencer control block by a host adapter integrated circuit	710/100
122	US 5562181 A	☒	Apparatus and method for automatically performing engine fluid changes	184/1.5
123	US 5559976 A	☒	System for instruction completion independent of result write-back responsive to both exception free completion of execution and completion of all logically prior instructions	712/215
124	US 5557764 A	☒	Interrupt vector method and apparatus	710/269
125	US 5553301 A	☒	Programmable sequencher having internal components which are microprocessor read/write interfacable	710/5
126	US 5548793 A	☒	System for controlling arbitration using the memory request signal types generated by the plurality of datapaths	710/40
127	US 5548738 A	☒	System and method for processing an instruction in a processing system	712/215

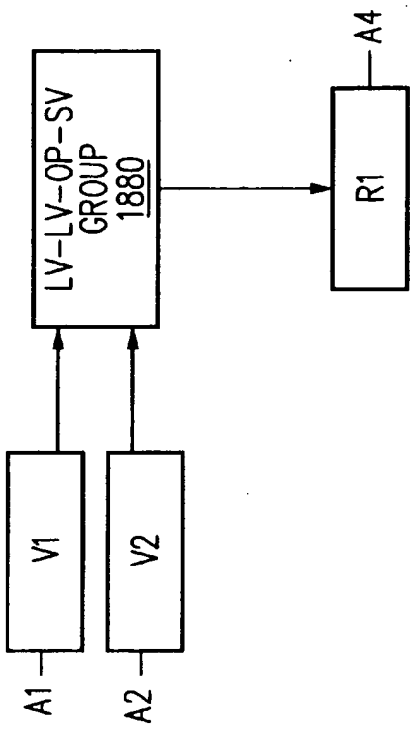

FIG. 18f

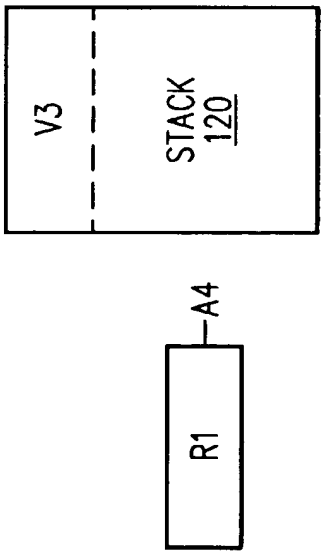

FIG. 18e

	Document ID	U	Title	Current OR
128	US 5542058 A	☒	Pipelined computer with operand context queue to simplify context-dependent execution flow	713/502
129	US 5530370 A	☒	Testing apparatus for testing and handling a multiplicity of devices	324/754
130	US 5528589 A	☒	Distributed type packet switching system and a method of controlling a flow control execution	370/235
131	US 5526500 A	☒	System for operand bypassing to allow a one and one-half cycle cache memory access time for sequential load and branch instructions	712/218
132	US 5522080 A	☒	Centralized control SIMD processor having different priority levels set for each data transfer request type and successively repeating the servicing of data transfer request in a predetermined order	710/220
133	US 5517665 A	☒	System for controlling arbitration using the memory request signal types generated by the plurality of datapaths having dual-ported local memory architecture for simultaneous data transmission	712/9
134	US 5515052 A	☒	Universal remote control with function synthesis	341/176
135	US 5502826 A	☒	System and method for obtaining parallel existing instructions in a particular data processing configuration by compounding instructions	712/213
136	US 5500942 A	☒	Method of indicating parallel execution compoundability of scalar instructions based on analysis of presumed instructions	712/210
137	US 5499350 A	☒	Vector data processing system with instruction synchronization	712/220
138	US 5495578 A	☒	Apparatus and method for changing the behavior of a computer program while retaining control of program execution	714/51
139	US 5485466 A	☒	Sequence controller including error correction and method therefor	700/21
140	US 5473763 A	☒	Interrupt vector method and apparatus	712/244
141	US 5473572 A	☒	Power saving system for a memory controller	365/227
142	US 5469572 A	☒	Post compile optimizer for linkable object code	717/152
143	US 5467468 A	☒	Semiconductor memory device having built-in test circuits selectively activated by decoder circuit	714/726
144	US 5457651 A	☒	Method and architecture for accelerated programming of uniform data into an electrically programmable memory	365/185.04
145	US 5452101 A	☒	Apparatus and method for decoding fixed and variable length encoded data	382/234
146	US 5448746 A	☒	System for comounding instructions in a byte stream prior to fetching and identifying the instructions for execution	712/210
147	US 5430854 A	☒	Simd with selective idling of individual processors based on stored conditional flags, and with consensus among all flags used for conditional branching	712/236
148	US 5428802 A	☒	Method and apparatus for executing critical disk access commands	710/9
149	US 5412784 A	☒	Apparatus for parallelizing serial instruction sequences and creating entry points into parallelized instruction sequences at places other than beginning of particular parallelized instruction sequence	712/245

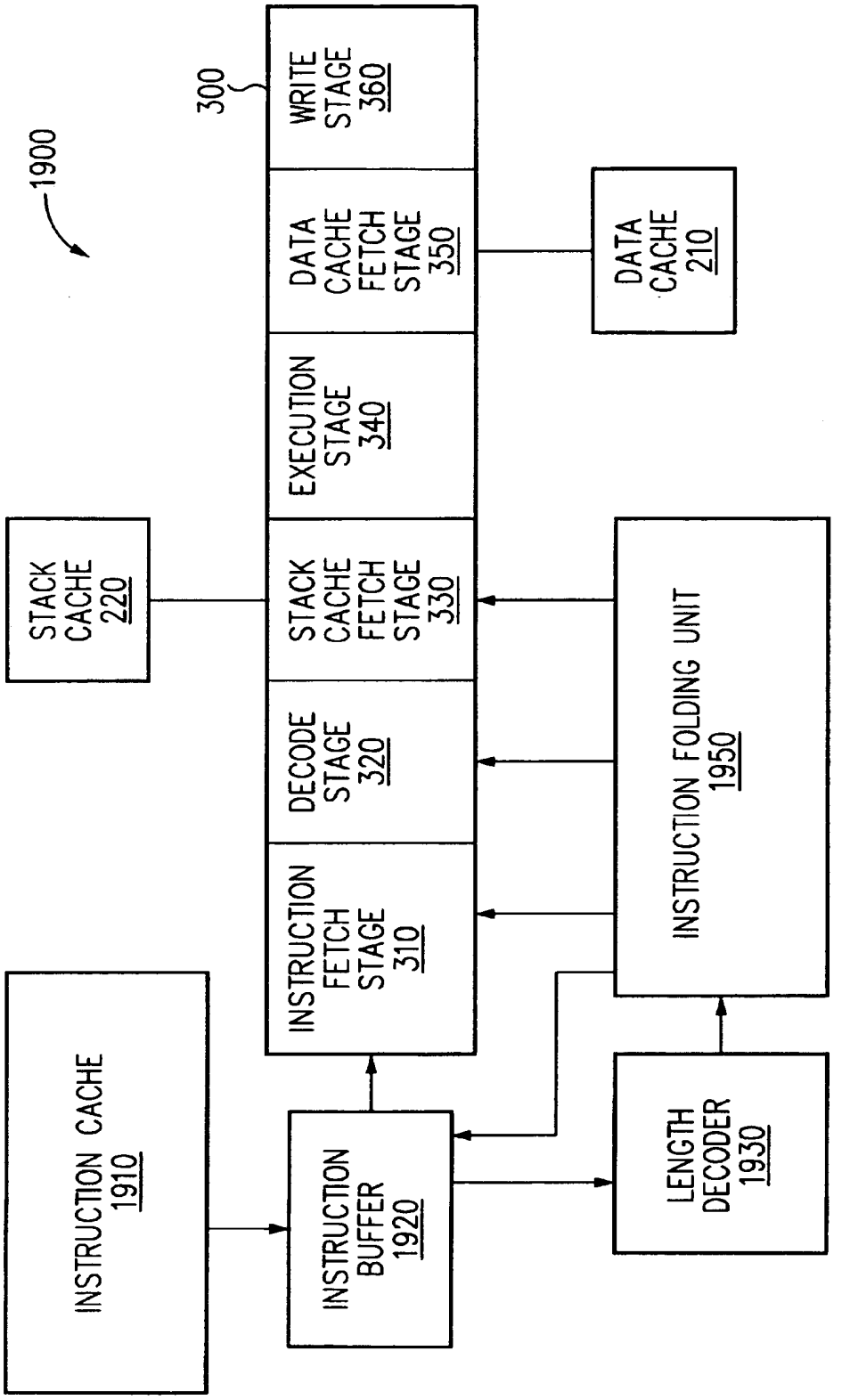
1900
300
WRITE STAGE 360
DATA CACHE FETCH STAGE 350
EXECUTION STAGE 340
STACK CACHE FETCH STAGE 330
DECODE STAGE 320
INSTRUCTION FETCH STAGE 310
DATA CACHE 210
STACK CACHE 220
INSTRUCTION FOLDING UNIT 1950
INSTRUCTION CACHE 1910
INSTRUCTION BUFFER 1920
LENGTH DECODER 1930

FIG. 19

	Document ID	U	Title	Current OR
150	US 5408658 A	☒	Self-scheduling parallel computer system and method	712/216
151	US 5394529 A	☒	Branch prediction unit for high-performance processor	712/240
152	US 5387928 A	☒	Electronic endoscope system having both still and moving images	348/70
153	US 5381419 A	☒	Method and apparatus for detecting retention faults in memories	714/720
154	US 5363296 A	☒	Electronic cash register having macro-keys	705/18
155	US 5347639 A	☒	Self-parallelizing computer system and method	712/203
156	US 5333296 A	☒	Combined queue for invalidates and return data in multiprocessor system	711/171
157	US 5317720 A	☒	Processor system with writeback cache using writeback and non writeback transactions stored in separate queues	711/143
158	US 5307504 A	☒	System and method for preserving instruction granularity when translating program code from a computer having a first architecture to a computer having a second reduced architecture during the occurrence of interrupts due to asynchronous events	712/41
159	US 5263153 A	☒	Monitoring control flow in a microprocessor	714/51
160	US 5261113 A	☒	Apparatus and method for single operand register array for vector and scalar data processing operations	712/8
161	US 5253186 A	☒	Process facility monitoring method including transformation of sequential conditions into constraining conditions	702/182
162	US 5239476 A	☒	Multi-level state language controller for multi-threaded machine control	700/159
163	US 5218712 A	☒	Providing a data processor with a user-mode accessible mode of operations in which the processor performs processing operations without interruption	710/261
164	US 5202889 A	☒	Dynamic process for the generation of biased pseudo-random test patterns for the functional verification of hardware designs	714/739
165	US 5185871 A	☒	Coordination of out-of-sequence fetching between multiple processors using re-execution of instructions	712/205
166	US 5185574 A	☒	NMR measurements using recursive RF excitation	324/309
167	US 5167035 A	☒	Transferring messages between nodes in a network	714/4
168	US 5155843 A	☒	Error transition mode for multi-processor system	714/5
169	US 5150977 A	☒	Recording apparatus with detector for paper edge and end of ribbon sensing	400/703
170	US 5150469 A	☒	System and method for processor pipeline control by selective signal deassertion	712/244
171	US 5114681 A	☒	Superfusion apparatus	422/111

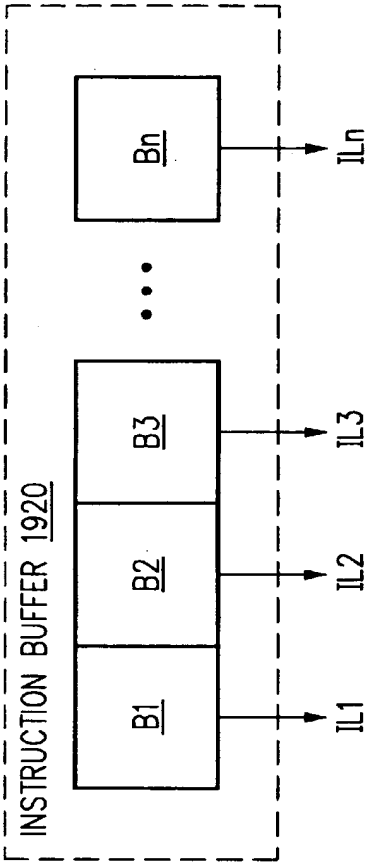
INSTRUCTION BUFFER 1920
B1
B2
B3
Bn
IL1
IL2
IL3
ILn

FIG. 20

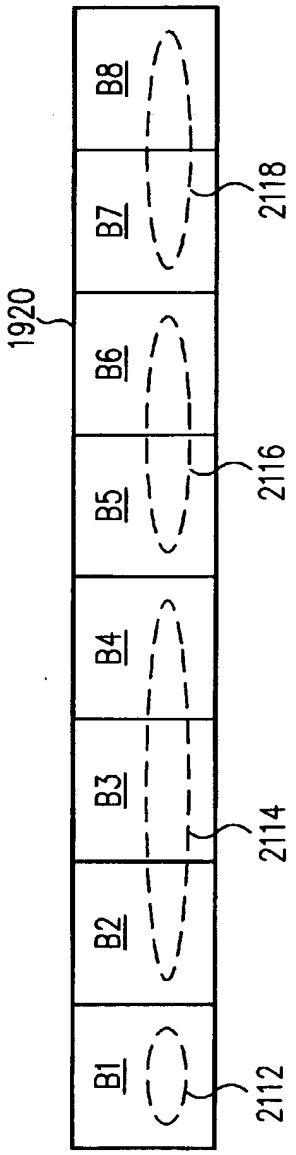
1920
B1
B2
B3
B4
B5
B6
B7
B8
2112
2114
2116
2118

FIG. 21

	Document ID	U	Title	Current OR
172	US 5113515 A	☒	Virtual instruction cache system using length responsive decoded instruction shifting and merging with prefetch buffer outputs to fill instruction buffer	711/125
173	US 5088048 A	☒	Massively parallel propositional reasoning	706/51
174	US 5058114 A	☒	Program control apparatus incorporating a trace function	714/38
175	US 5023728 A	☒	Image forming apparatus	358/437
176	US 4989172 A	☒	Apparatus and method for checking start signals	708/530
177	US 4956800 A	☒	Arithmetic operation processing apparatus of the parallel processing type and compiler which is used in this apparatus	708/524
178	US 4956770 A	☒	Method and device to execute two instruction sequences in an order determined in advance	712/220
179	US 4953078 A	☒	Apparatus and method for multi-threaded program execution in a microcoded data processing system	712/248
180	US 4947369 A	☒	Microword generation mechanism utilizing a separate branch decision programmable logic array	712/234
181	US 4943916 A	☒	Information processing apparatus for a data flow computer	712/26
182	US 4926323 A	☒	Streamlined instruction processor	712/238
183	US 4926308 A	☒	Programmable machine system	700/18
184	US 4916602 A	☒	Microcode control system in a microcomputer capable of branching to different microcode routines	712/245
185	US 4868745 A	☒	Data processing system and method for the direct and indirect execution of uniformly structured object types	711/202
186	US 4868735 A	☒	Interruptible structured microprogrammed sixteen-bit address sequence controller	712/234
187	US 4853889 A	☒	Arrangement and method for speeding the operation of branch instructions	712/237
188	US 4811211 A	☒	On-line overflow response system and ALU branching structure	708/525
189	US 4802116 A	☒	Programmed controller	703/23
190	US 4742453 A	☒	Pipeline-controlled information processing system for generating updated condition code	712/234
191	US 4719565 A	☒	Interrupt and trap handling in microprogram sequencer	710/260
192	US 4695983 A	☒	Calculator or pocket computer with selectable operational sequence	708/144
193	US 4686464 A	☒	Digital interface subsystem for a magnetic resonance imaging and spectroscopy system	324/309
194	US 4586127 A	☒	Multiple control stores for a pipelined microcontroller	712/243

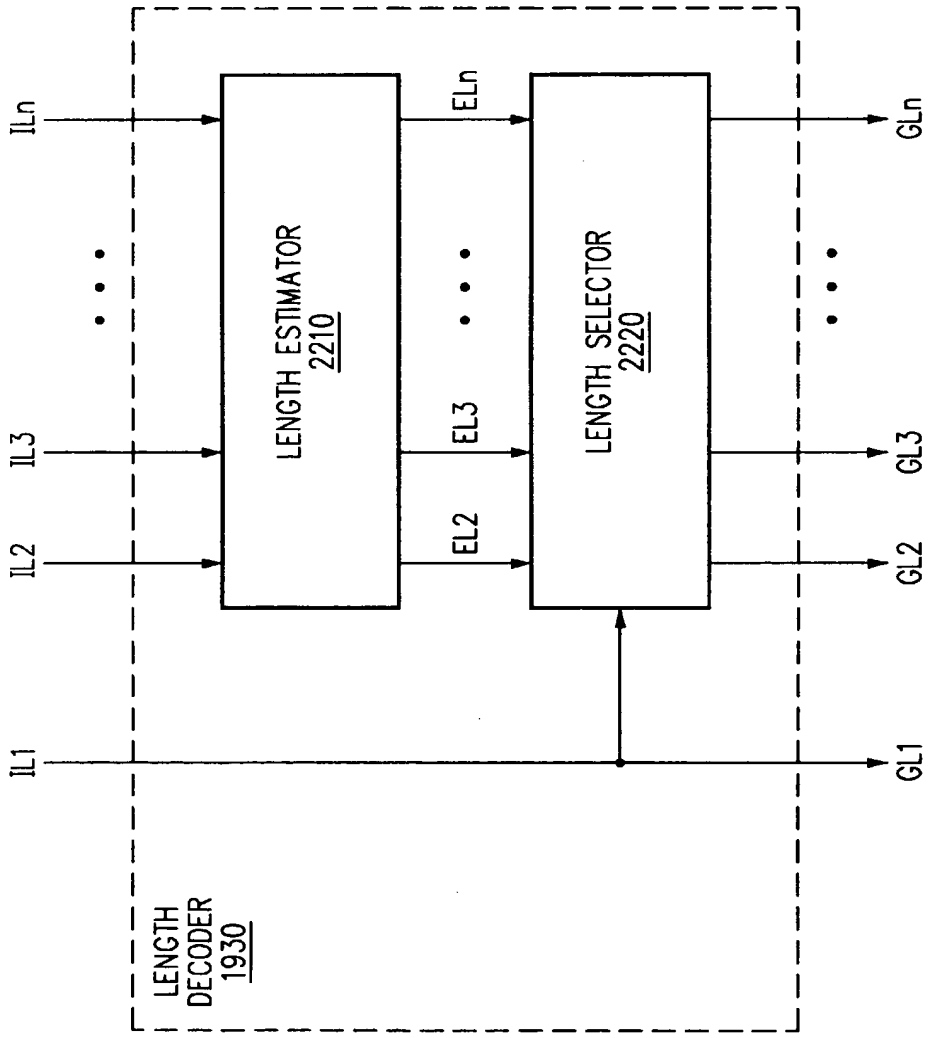
ILn
IL3
IL2
IL1
LENGTH ESTIMATOR
2210
ELn
EL3
EL2
LENGTH SELECTOR
2220
GLn
GL3
GL2
GL1
LENGTH
DECODER
1930

FIG. 22

	Document ID	U	Title	Current OR
195	US 4575816 A	☒	Interactive transactions processor using sequence table pointers to access function table statements controlling execution of specific interactive functions	345/853
196	US 4551798 A	☒	Multiple control stores in a pipelined microcontroller for handling nester subroutines	712/243
197	US 4547860 A	☒	Computer keyboards with few keys designating hundreds of functions	708/146
198	US 4546431 A	☒	Multiple control stores in a pipelined microcontroller for handling jump and return subroutines	712/243
199	US 4521850 A	☒	Instruction buffer associated with a cache memory unit	712/200
200	US 4503528 A	☐	Method and apparatus for controlling vehicle mounted devices and equipment	367/198

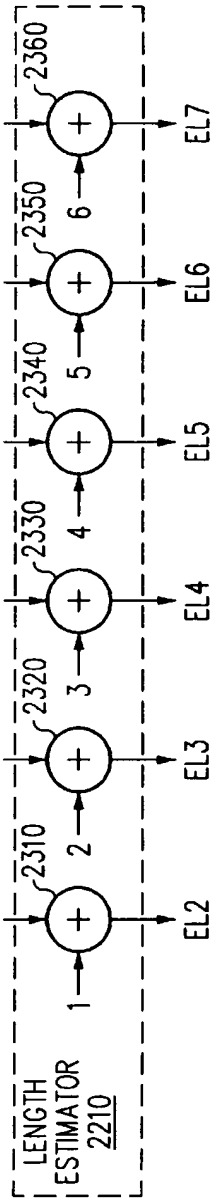
LENGTH ESTIMATOR 2210
BL2
BL3
BL4
BL5
BL6
BL7
2310
2320
2330
2340
2350
2360
1
2
3
4
5
6
EL2
EL3
EL4
EL5
EL6
EL7

FIG. 23

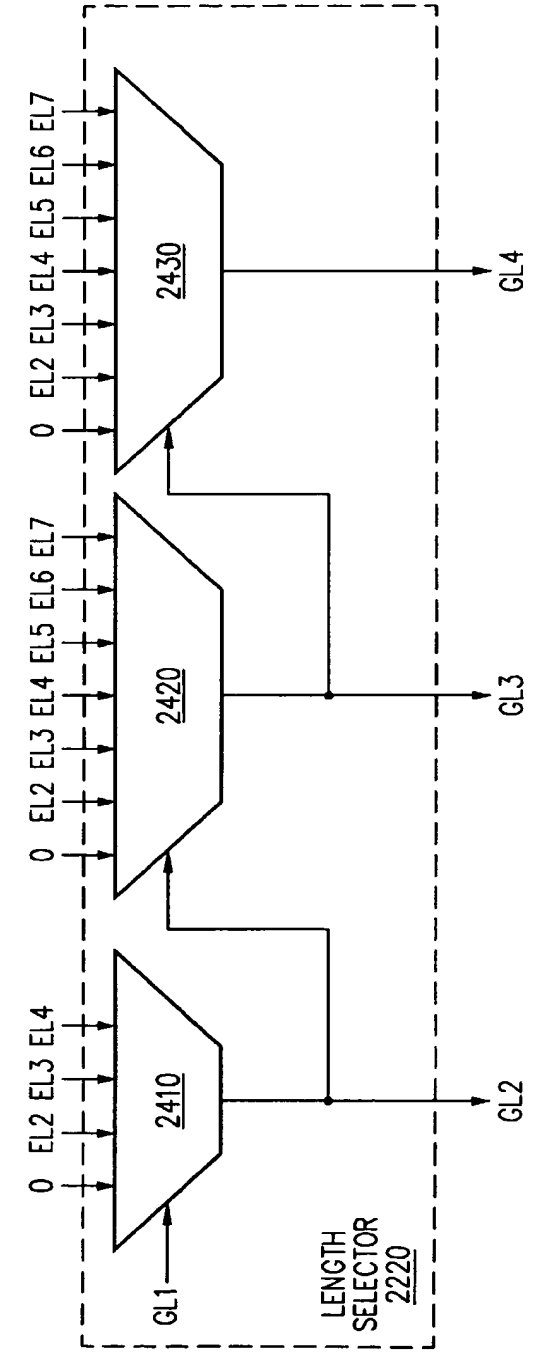
LENGTH SELECTOR 2220
0 EL2 EL3 EL4
0 EL2 EL3 EL4 EL5 EL6 EL7
0 EL2 EL3 EL4 EL5 EL6 EL7
2410
2420
2430
GL1
GL2
GL3
GL4

FIG. 24

	L #	Hits	Search Text	DBs
1	L1	93761	(speculative$3 sequen$4 order) near10 (instruction execut$3)	USPAT; US-PGPUB
2	L2	1692	(indicat$3 flag tag bit field mode) near10 (allow$3 permit$4 enabl$3 disabl$3 prohibit$3 inhibit$3 activ$5 deactiv$5) near20 1	USPAT; US-PGPUB
3	L3	445	((speculative$3 sequen$4 order) and (instruction execut$3)).ab,ti. and 2	USPAT; US-PGPUB
4	L8	95938	(speculative$3 sequen$4 order predict$3) near10 (instruction execut$3)	USPAT; US-PGPUB
5	L9	1793	(indicat$3 flag tag bit field mode) near10 (allow$3 permit$4 enabl$3 disabl$3 prohibit$3 inhibit$3 activ$5 deactiv$5) near20 8	USPAT; US-PGPUB
6	L10	527	((speculative$3 sequen$4 order predict$3) and (instruction execut$3)).ab,ti. and 9	USPAT; US-PGPUB
7	L11	31579	(speculative$3 sequen$4 order predict$3) near10 (instruction execut$3)	EPO; JPO; DERWENT; IBM_TDB
8	L12	157	(indicat$3 flag tag bit field mode) near10 (allow$3 permit$4 enabl$3 disabl$3 prohibit$3 inhibit$3 activ$5 deactiv$5) near20 11	EPO; JPO; DERWENT; IBM_TDB
9	L13	62	9 near10 speculative$3	USPAT; US-PGPUB
10	L19	95947	(speculat$6 sequen$4 order predict$3) near10 (instruction execut$3)	USPAT; US-PGPUB
11	L22	31612	(speculat$6 sequen$4 order predict$3) near10 (instruction execut$3)	EPO; JPO; DERWENT; IBM_TDB
12	L33	1734	(indicat$3 flag tag bit field mode) near10 ((allow$3 permit$4 enabl$3 disabl$3 prohibit$3 inhibit$3 activ$5 deactiv$5) near10 19)	USPAT; US-PGPUB
13	L34	563	((speculat$6 sequen$4 order predict$3 load) and (instruction execut$3)).ab,ti. and 33	USPAT; US-PGPUB
14	L35	25	(indicat$3 flag tag bit field mode) near10 ((allow$3 permit$4 enabl$3 disabl$3 prohibit$3 inhibit$3 activ$5 deactiv$5) near10 22) not 12	EPO; JPO; DERWENT; IBM_TDB
15	L36	56467	(speculat$6 order predict$3) near10 (instruction execut$3)	USPAT; US-PGPUB
16	L37	273	(indicat$3 flag tag bit field mode) near10 ((allow$3 permit$4 enabl$3 disabl$3 prohibit$3 inhibit$3 activ$5 deactiv$5) near10 36) and 34 not 13	USPAT; US-PGPUB
17	L38	243	34 not (37 13)	USPAT; US-PGPUB

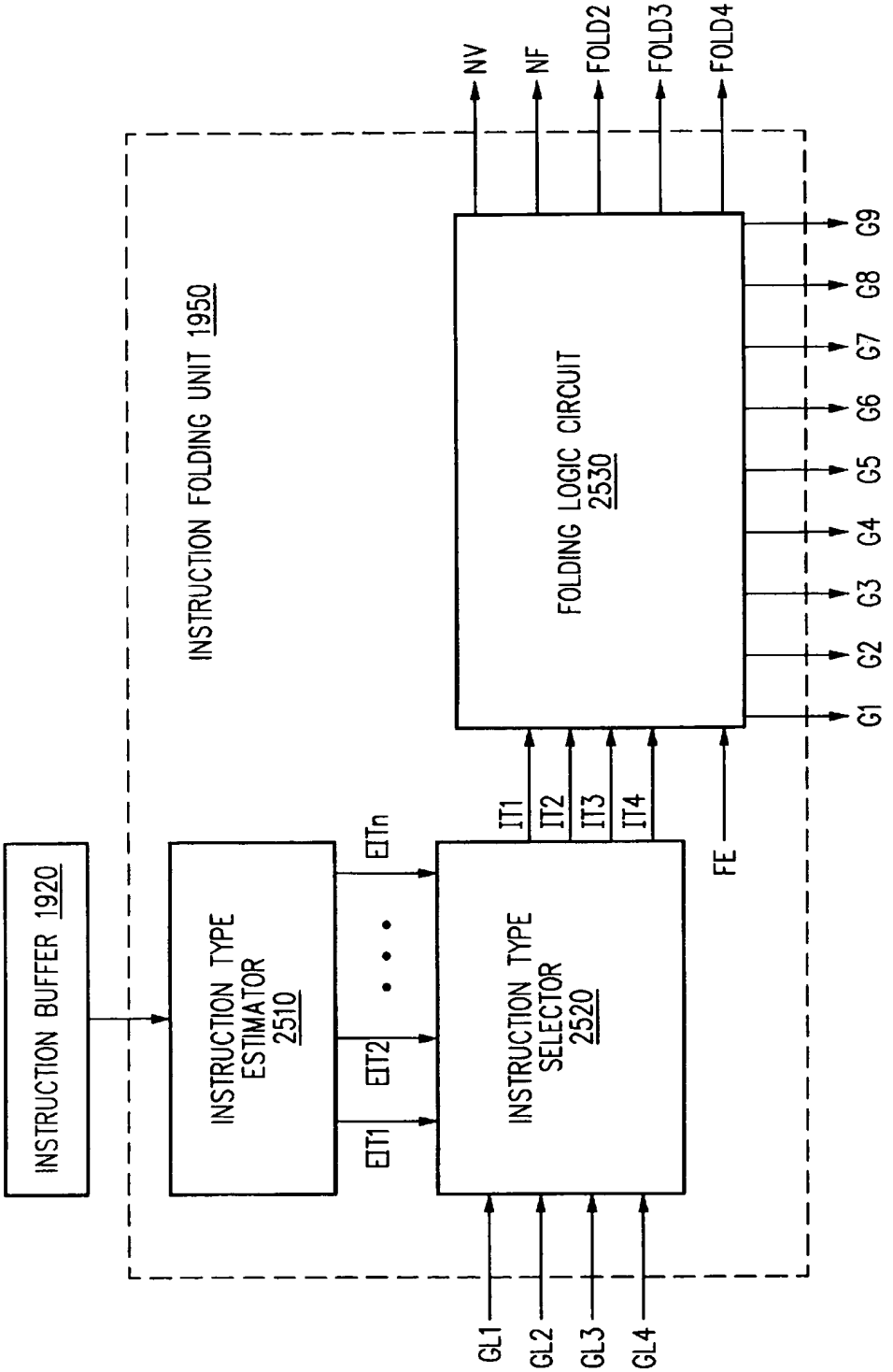
INSTRUCTION BUFFER 1920
INSTRUCTION FOLDING UNIT 1950
INSTRUCTION TYPE ESTIMATOR 2510
EITn
EIT2
EIT1
INSTRUCTION TYPE SELECTOR 2520
IT1
IT2
IT3
IT4
FE
FOLDING LOGIC CIRCUIT 2530
NV
NF
FOLD2
FOLD3
FOLD4
G9
G8
G7
G6
G5
G4
G3
G2
G1
GL1
GL2
GL3
GL4

FIG. 25

	L #	Hits	Search Text	DBs
1	L1	93761	(speculative$3 sequen$4 order) near10 (instruction execut$3)	USPAT; US-PGPUB
2	L2	1692	(indicat$3 flag tag bit field mode) near10 (allow$3 permit$4 enabl$3 disabl$3 prohibit$3 inhibit$3 activ$5 deactiv$5) near20 1	USPAT; US-PGPUB
3	L3	445	((speculative$3 sequen$4 order) and (instruction execut$3)).ab,ti. and 2	USPAT; US-PGPUB
4	L8	95938	(speculative$3 sequen$4 order predict$3) near10 (instruction execut$3)	USPAT; US-PGPUB
5	L9	1793	(indicat$3 flag tag bit field mode) near10 (allow$3 permit$4 enabl$3 disabl$3 prohibit$3 inhibit$3 activ$5 deactiv$5) near20 8	USPAT; US-PGPUB
6	L10	527	((speculative$3 sequen$4 order predict$3) and (instruction execut$3)).ab,ti. and 9	USPAT; US-PGPUB
7	L11	31579	(speculative$3 sequen$4 order predict$3) near10 (instruction execut$3)	EPO; JPO; DERWENT; IBM_TDB
8	L12	157	(indicat$3 flag tag bit field mode) near10 (allow$3 permit$4 enabl$3 disabl$3 prohibit$3 inhibit$3 activ$5 deactiv$5) near20 11	EPO; JPO; DERWENT; IBM_TDB
9	L13	62	9 near10 speculative$3	USPAT; US-PGPUB
10	L19	95947	(speculat$6 sequen$4 order predict$3) near10 (instruction execut$3)	USPAT; US-PGPUB
11	L20	1797	(indicat$3 flag tag bit field mode) near10 (allow$3 permit$4 enabl$3 disabl$3 prohibit$3 inhibit$3 activ$5 deactiv$5) near20 19	USPAT; US-PGPUB
12	L22	31612	(speculat$6 sequen$4 order predict$3) near10 (instruction execut$3)	EPO; JPO; DERWENT; IBM_TDB
13	L23	0	(indicat$3 flag tag bit field mode) near10 (allow$3 permit$4 enabl$3 disabl$3 prohibit$3 inhibit$3 activ$5 deactiv$5) near20 22 not 12	EPO; JPO; DERWENT; IBM_TDB
14	L25	529	((speculat$6 sequen$4 order predict$3) and (instruction execut$3)).ab,ti. and 20	USPAT; US-PGPUB
15	L26	487	25 not 13	USPAT; US-PGPUB

g

Printed by HPS Server
for

Walk-Up_Printing

Printer: cpk2_2c21_gblrptr
Date: 03/12/04
Time: 16:33:53

Document Listing

Document	Selected Pages	Page Range	Copies
US006499098	20	1 - 20	1
US006330657	21	1 - 21	1
US005163139	12	1 - 12	1
Total (3)	53	-	-

	Document ID	U	Title	Current OR
1	US 5560032 A	☐	High-performance, superscalar-based computer system with out-of-order instruction execution and concurrent results distribution	712/23
2	US 5560025 A	☒	Entry allocation apparatus and method of same	712/23
3	US 5555428 A	☒	Activity masking with mask context of SIMD processors	712/22
4	US 5553255 A	☒	Data processor with programmable levels of speculative instruction fetching and method of operation	712/235
5	US 5539911 A	☒	High-performance, superscalar-based computer system with out-of-order instruction execution	712/23
6	US 5530815 A	☒	Apparatus and method for verifying the order and operation of a data processing device when asynchronous commands are held in a command queue	712/227
7	US 5524088 A	☒	Multi-functional operating circuit providing capability of freely combining operating functions	708/493
8	US 5522053 A	☒	Branch target and next instruction address calculation in a pipeline processor	711/213
9	US 5507028 A	☒	History based branch prediction accessed via a history based earlier instruction address	712/207
10	US 5500947 A	☒	Operand specifier processing by grouping similar specifier types together and providing a general routine for each	712/211
11	US 5490280 A	☒	Apparatus and method for entry allocation for a resource buffer	712/23
12	US 5481744 A	☒	Microcode sequencer changing states in response to an external gating input level change upon the occurrence of a wait instruction	712/227
13	US 5481679 A	☒	Data processing apparatus having bus switches for selectively connecting buses to improve data throughput	710/316
14	US 5471593 A	☒	Computer processor with an efficient means of executing many instructions simultaneously	712/235
15	US 5465361 A	☒	Microcode linker/loader that generates microcode sequences for MRI sequencer by modifying previously generated microcode sequences	717/168
16	US 5463432 A	☒	Miniature pan/tilt tracking mount	352/243
17	US 5454117 A	☒	Configurable branch prediction for a processor performing speculative execution	712/23
18	US 5454086 A	☒	Dynamic program analyzer facility	712/227
19	US 5454077 A	☒	Communication system between a plurality of transmitters and receivers having relays responsive to those identifying codes of transmitters contained in its respective table memory	710/106
20	US 5448705 A	☒	RISC microprocessor architecture implementing fast trap and exception state	712/244
21	US 5443427 A	☒	Apparatus for controlling automatic transmission	475/123
22	US 5437048 A	☒	Programmable controller acting as a master station and having automatic control of interlock process by using an operation complete address flag	709/208
23	US 5414822 A	☒	Method and apparatus for branch prediction using branch prediction table with improved branch prediction effectiveness	712/240

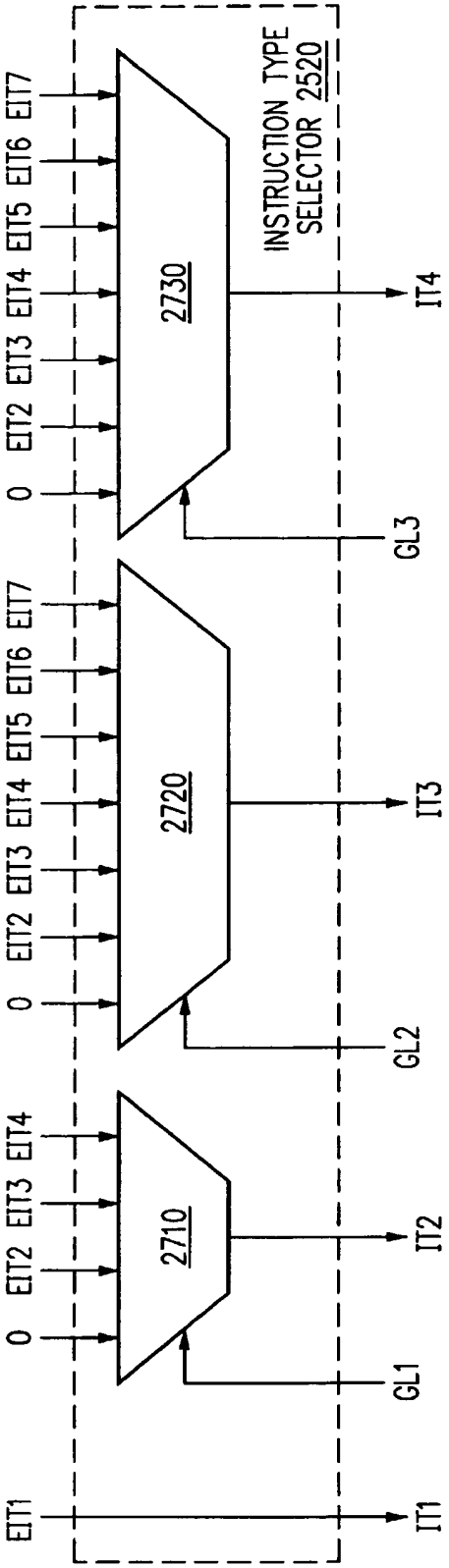
INSTRUCTION TYPE SELECTOR 2520
2730
2720
2710
EIT1
EIT2
EIT3
EIT4
EIT5
EIT6
EIT7
0
GL1
GL2
GL3
IT1
IT2
IT3
IT4

FIG. 27

	Document ID	U	Title	Current OR
24	US 5313644 A	☒	System having status update controller for determining which one of parallel operation results of execution units is allowed to set conditions of shared processor status word	712/228
25	US 5295248 A	☒	Branch control circuit	712/239
26	US 5291610 A	☒	Microcode sequencer changing states in response to an external gating input level change upon the occurrence of a WAIT instruction	712/226
27	US 5287467 A	☒	Pipeline for removing and concurrently executing two or more branch instructions in synchronization with other instructions executing in the execution unit	712/235
28	US 5247693 A	☒	Computer language structure for process control applications and method of translating same into program code to operate the computer	717/139
29	US 5228131 A	☒	Data processor with selectively enabled and disabled branch prediction operation	712/240
30	US 5206938 A	☒	IC card with memory area protection based on address line restriction	711/200
31	US 5193156 A	☒	Data processor with pipeline which disables exception processing for non-taken branches	712/239
32	US 5168566 A	☒	Multi-task control device for central processor task execution control provided as a peripheral device and capable of prioritizing and timesharing the tasks	718/103
33	US 5159676 A	☒	Semi-smart DRAM controller IC to provide a pseudo-cache mode of operation using standard page mode draws	711/107
34	US 5144242 A	☒	Continually loadable microcode store for MRI control sequencers	324/312
35	US 5101353 A	☒	Automated system for providing liquidity to securities markets	705/37
36	US 5101341 A	☒	Pipelined system for reducing instruction access time by accumulating predecoded instruction bits a FIFO	712/213
37	US 5081574 A	☒	Branch control in a three phase pipelined signal processor	712/234
38	US 5072364 A	☒	Method and apparatus for recovering from an incorrect branch prediction in a processor that executes a family of instructions in parallel	712/215
39	US 5008807 A	☒	Data processing apparatus with abbreviated jump field	712/213
40	US 4939644 A	☒	Input/output controller for controlling the sequencing of the execution of input/output commands in a data processing system	710/5
41	US 4901232 A	☒	I/O controller for controlling the sequencing of execution of I/O commands and for permitting modification of I/O controller operation by a host processor	710/6
42	US 4870573 A	☒	Microcomputer capable of testing execution of a program with no branch	714/38
43	US 4868740 A	☒	System for processing data with multiple virtual address and data word lengths	711/2
44	US 4860199 A	☒	Hashing indexer for branch cache	711/213
45	US 4858104 A	☒	Preceding instruction address based branch prediction in a pipelined processor	712/240
46	US 4853840 A	☒	Instruction prefetching device including a circuit for checking prediction of a branch instruction before the instruction is executed	712/237

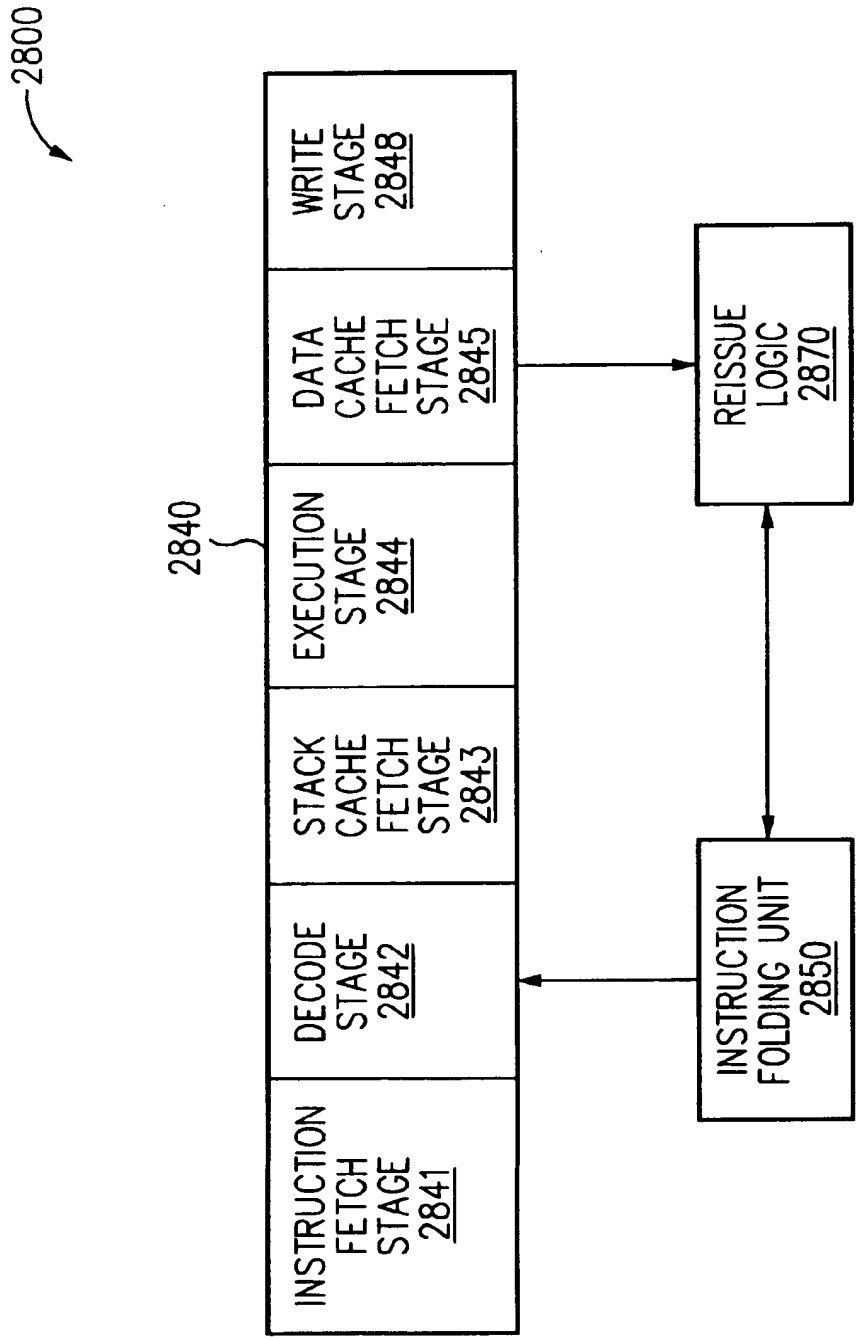
2800
2840
INSTRUCTION FETCH STAGE 2841
DECODE STAGE 2842
STACK CACHE FETCH STAGE 2843
EXECUTION STAGE 2844
DATA CACHE FETCH STAGE 2845
WRITE STAGE 2848
REISSUE LOGIC 2870
INSTRUCTION FOLDING UNIT 2850

FIG. 28

	Document ID	U	Title	Current OR
47	US 4794517 A	☒	Three phased pipelined signal processor	712/32
48	US 4785393 A	☒	32-Bit extended function arithmetic-logic unit on a single chip	712/221
49	US 4777594 A	☒	Data processing apparatus and method employing instruction flow prediction	712/240
50	US 4760519 A	☒	Data processing apparatus and method employing collision detection and prediction	712/217
51	US 4755935 A	☒	Prefetch memory system having next-instruction buffer which stores target tracks of jumps prior to CPU access of instruction	712/233
52	US 4750112 A	☒	Data processing apparatus and method employing instruction pipelining	712/217
53	US 4741311 A	☒	Method of air/fuel ratio control for internal combustion engine	123/675
54	US 4736320 A	☒	Computer language structure for process control applications, and translator therefor	717/109
55	US 4688188 A	☒	Data storage apparatus for storing groups of data with read and write request detection	711/114
56	US 4672360 A	☒	Apparatus and method for converting a number in binary format to a decimal format	341/104
57	US 4594661 A	☒	Microword control system utilizing multiplexed programmable logic arrays	712/248
58	US 4557694 A	☒	Teaching device and method of using same	434/339
59	US 4484300 A	☒	Data processor having units carry and tens carry apparatus supporting a decimal multiply operation	708/623
60	US 4462073 A	☒	Apparatus for fetching and decoding instructions	712/207
61	US 4456994 A	☒	Remote simulation by remote control from a computer desk	714/33
62	US 4447877 A	☒	Memory bus interface system	713/502
63	US 4430706 A	☒	Branch prediction apparatus and method for a data processing system	712/237
64	US 4426680 A	☒	Data processor using read only memories for optimizing main memory access and identifying the starting position of an operand	711/214
65	US 4384341 A	☒	Data processor having carry apparatus supporting a decimal divide operation	708/652
66	US 4316244 A	☒	Memory apparatus for digital computer system	711/168
67	US 4247894 A	☒	Arrangement for program interruption	710/264
68	US 4040021 A	☒	Circuit for increasing the apparent occupancy of a processor	713/601
69	US 4037094 A	☒	Multi-functional arithmetic and logical unit	708/231

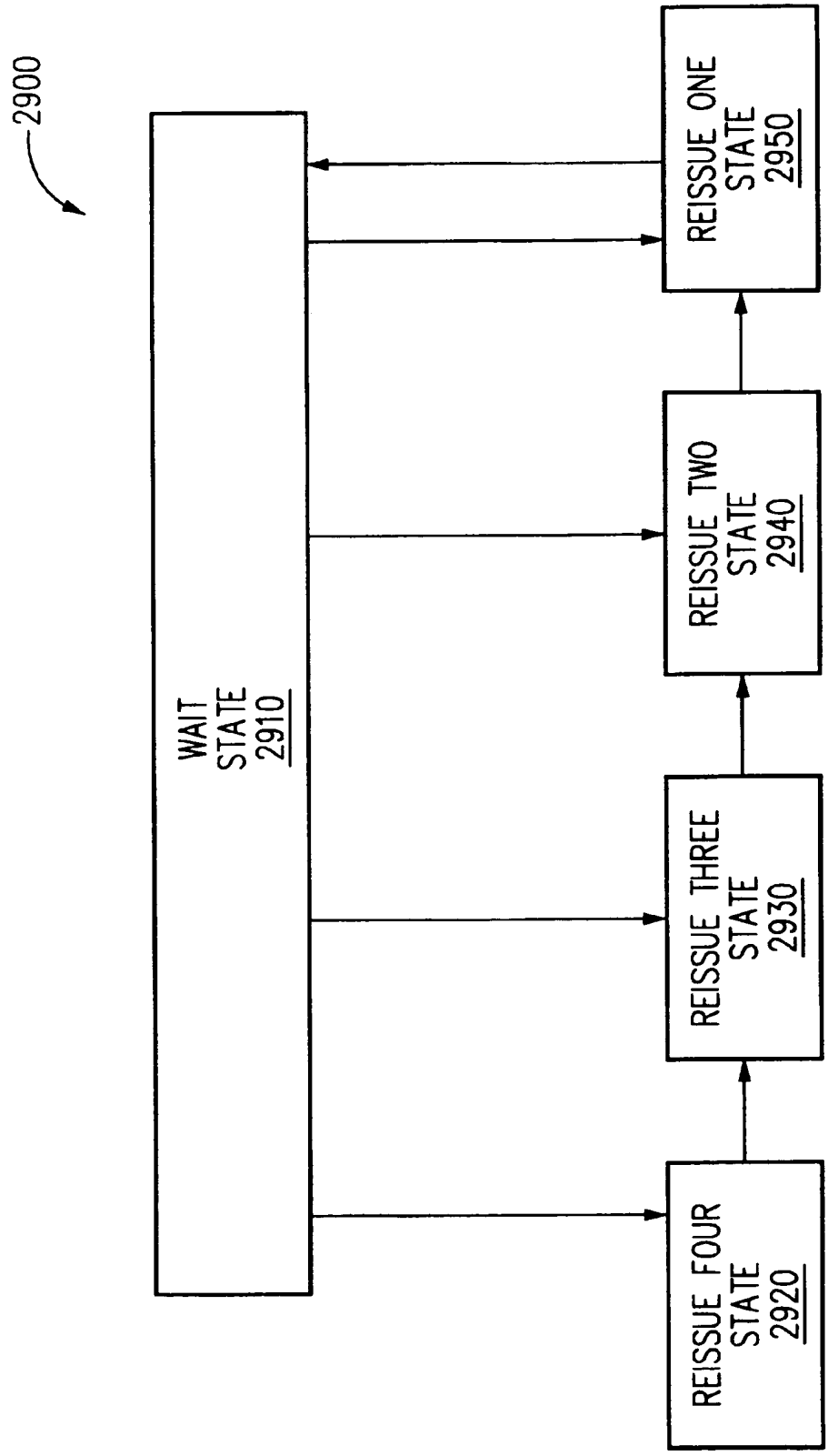
2900
WAIT STATE 2910
REISSUE FOUR STATE 2920
REISSUE THREE STATE 2930
REISSUE TWO STATE 2940
REISSUE ONE STATE 2950

FIG. 29

	Document ID	U	Title	Current OR
70	US 3962684 A	☒	Computing system interface using common parallel bus and segmented addressing	710/100
71	US 3962683 A	☒	CPU programmable control system	713/600
72	US 3825903 A	☒	AUTOMATIC SWITCHING OF STORAGE PROTECT KEYS	711/163
73	US 3757306 A	☒	COMPUTING SYSTEMS CPU	712/32
74	US 3736567 A	☒	PROGRAM SEQUENCE CONTROL	712/231

LENGTH DECODER FOR VARIABLE LENGTH DATA

CROSS-REFERENCE TO RELATED APPLICATIONS

This application relates to the co-pending application Ser. No. 09/064,807, filed Apr. 22, 1998, "SUPERSCALAR STACK BASED COMPUTING SYSTEM", by Koppala, et. al. owned by the assignee of this application and incorporated herein by reference.

This application relates to the co-pending application Ser. No. 09/064,682, filed Apr. 22, 1998, "REISSUE LOGIC FOR HANDLING TRAPS IN A MULTIISSUE STACK BASED COMPUTING SYSTEM", by Koppala, et. al. owned by the assignee of this application and incorporated herein by reference.

This application relates to the co-pending application Ser. No. 09/064,686, filed Apr. 22, 1998, "STACK CACHE MISS HANDLING", by Koppala, et. al. owned by the assignee of this application and incorporated herein by reference.

BACKGROUND OF THE INVENTION

1. Field of the Invention

The present invention relates generally to computing systems and, in particular, to super-scalar stack based computing systems.

2. Discussion of Related Art

Most computing systems are coupled to a random access memory system for storing and retrieving data. Various ways to increase the speed of computing systems using random access memory systems are well known in the art. For example using caches between a central processing unit of a computing system and the memory system can improve memory throughput. Furthermore, super-scalar architectures and pipelining can improve the performance of central processing units.

However, other memory architectures such as stacks are also used in computing systems. As shown in FIG. **1**, a stack based computing system **110**, which can implement for example, the JAVA Virtual Machine, is coupled to a stack **120**. In classical stack architectures, data is either "pushed" onto the stack or "popped" off the stack by stack based computing system **110**. For example, to add the numbers 4 and 5, a stack based computing system **110** first pushes the number 4 onto the top of stack **120**. Then, stack based computing system **110** pushes the number 5 onto the stack. Then, stack based computing system **110** performs an add operation which pops the number 5 off stack **120** and the number 4 off stack **120** and pushes the number 9 onto the top of stack **120**. A major advantage of stack based computing system **110** is that operations using data at the top of the stack do not need to use memory addresses. The top of stack is also referred to as the first location of the stack, and the location just under the top of the stack is also referred to as the second location of the stack. Similarly, the memory location in the stack just after the second location is also referred to as the third location of the stack.

Stack based computing system **110** can become more flexible by also allowing stack based computing system **110** to use some random access techniques with stack **120**. Thus, in some implementation of stack based computing system **110** and stack **120**, the memory locations in stack **120** are part of a random-access memory architecture. Thus, each memory location in stack **120** has a memory address. As used herein, a memory location having a memory address equal to x is referred to as memory location x.

Even in stack based computing systems using random-access techniques, most operations by the stack based computing system use data from or near the top of stack **120**. For example, assume a value **V1** from a memory location ADDR1 is to be added to a value **V2** from a memory location ADDR2, and the sum stored at a memory location ADDR3, stack based computing system **110** first executes a stack load instruction, which retrieves value **V1** from memory location ADDR1 and pushes value **V1** onto the top of stack **120**. Next, stack based computing system **110** executes another stack load instruction, which retrieves value **V2** from memory location ADDR2 and pushes value **V2** onto the top of stack **120**. Then, stack based computing system **110** executes an add instruction which pops the top two locations of stack **120**, which now contain value **V1** and value **V2**, and pushes the sum of value **V1** and value **V2** onto the top of stack **120**. Finally, stack based computing system **110** executes a stack store instruction which pops the value from the top of stack **120**, i.e. the sum of value **V1** and value **V2**, and stores the value in memory location ADDR3.

Some of the techniques used to improve the performance of random access memory systems can be adapted to improve stack performance. For example, as shown in FIG. **2**, stack **120** can contain a data cache **210**, a stack cache **220**, a stack cache management unit **240**, and a memory circuit **230**. Data cache **210** is formed with fast memory circuits, such as SRAMS, to improve the throughput of memory circuit **230**. Stack cache **220** specifically caches a top portion of stack **120** using fast memory circuits, such as SRAMS. Stack cache management unit **240** manages stack cache **220** by copying data from memory circuit **230** into stack cache **220** as data is popped off of stack **120** or spilling data from stack cache **220** to memory circuit **230** as data is pushed onto stack **120**. Thus, stack cache **220** maintains the top of stack **120** in fast memory circuit, so that a stack based computing system can perform stack operations with low stack latency. Specific implementations of stack caches and stack management units are described in U.S. patent application Ser. No. 08/828,899, entitled "Stack Caching Circuit with Overflow/Underflow unit", by Sailendra Koppala, which is hereby incorporated by reference.

Once stack latency is reduced, the operating speed of a stack based computing system may be limited by the rate at which stack operations can be performed. In general-purpose processing units, such as RISC microprocessors, pipelining and super-scalar implementation are used to improve the performance of the processing units. However, the techniques used for RISC processors are not easily adapted to stack based computing systems. For example, in super-scalar architectures, data dependencies determine which instructions can be issued simultaneously. However, for stack based computing systems, most stack operations use the top of the stack and would thus have a data dependency conflict. Hence, there is a need for a stack based computing system architecture to improve the performance of stack based computing systems.

SUMMARY

Accordingly, the present invention provides pipelining techniques to prevent pipeline stalls and a super-scalar architecture for stack based computing systems, which can issue multiple stack operations concurrently. In accordance with one embodiment of the present invention, a stack based computing system includes an instruction pipeline, which

	Document ID	U	Title	Current OR
1	US 20030 16729 2 A1	☐	Method and apparatus for performing critical tasks using speculative operations	718/101
2	US 20030 10133 5 A1	☐	Method for binary-level branch reversal on computer architectures supporting predicated execution	712/234
3	US 20030 08875 9 A1	☐	System and method to reduce execution of instructions involving unreliable data in a speculative processor	712/218
4	US 20030 03351 0 A1	☐	Methods and apparatus for controlling speculative execution of instructions based on a multiaccess memory condition	712/235
5	US 20020 19446 4 A1	☐	Speculative branch target address cache with selective override by seconday predictor based on branch instruction type	712/239
6	US 20020 19446 3 A1	☐	Speculative hybrid branch direction predictor	712/239
7	US 20020 19446 2 A1	☐	Apparatus and method for selecting one of multiple target addresses stored in a speculative branch target address cache per instruction cache line	712/238
8	US 20020 19446 1 A1	☐	Speculative branch target address cache	712/238
9	US 20020 19446 0 A1	☐	Apparatus, system and method for detecting and correcting erroneous speculative branch target address cache branches	712/238
10	US 20020 18883 4 A1	☐	Apparatus and method for target address replacement in speculative branch target address cache	712/238
11	US 20020 18883 3 A1	☐	Dual call/return stack branch prediction system	712/236
12	US 20020 09200 2 A1	☐	Method and apparatus for preserving precise exceptions in binary translated code	717/137
13	US 20020 06600 5 A1	☐	Data processor with an improved data dependence detector	712/218
14	US 20010 00475 7 A1	☐	Processor and method of controlling the same	712/218
15	US 66315 14 B1	☐	Emulation system that uses dynamic binary translation and permits the safe speculation of trapping operations	717/137
16	US 66292 38 B1	☐	Predicate controlled software pipelined loop processing with prediction of predicate writing and value prediction for use in subsequent iteration	712/241
17	US 66222 35 B1	☐	Scheduler which retries load/store hit situations	712/23
18	US 66153 40 B1	☐	Extended operand management indicator structure and method	712/209

instructions.

8. The digital computer system as in claim 7 wherein means coupling said input means of said instruction compounding mechanism to an output Of said larger-capacity, lower-speed storage mechanism for transferring instructions from said larger-capacity, lower-speed storage mechanism to said instruction compounding mechanism transfers said instructions in unmodified object code.

9. The combination of claim 8 wherein the instruction compounding mechanism includes:

a plural-instruction instruction register for receiving a plurality of successive instructions from the lower-speed storage mechanism;

a plurality of rule-based instruction analyzer mechanisms, each of which analyzes a different pair of side-by-side instructions i the instruction register and includes means for producing a compoundability signal which indicates whether or not th two instructions in its pair may be processed in parallel;

and a tag generating mechanism responsive to the compoundability signals for generating th individual tag fields for the analyzed instructions in the instruction register.

10. The combination of claim 9 wherein:

the computer system has a particular instruction processing configuration having a plurality of functional units of one or more types;

and each instruction analyzer mechanism includes logic circuitry for implementing rules which define which types of instructions are compatible for parallel execution in the particular instruction processing configuration used for the computer system, such logic circuitry providing said means for producing the compoundability signal for that analyzer mechanism.

11. The digital computer system as in claim 1, wherein the instruction compounding mechanism includes:

a plural-instruction instruction register for receiving a plurality of successive instructions from the lower-speed storage mechanism;

a plurality of rule-based instruction analyzer mechanisms, each of which analyzes a different pair of side-by-side instructions with unmodified object code in the instruction register and produces a compoundability signal which indicates whether or not the two instructions in its pair may be processed in parallel; and

a tag generating mechanism responsive to the compoundability signals for generating the individual tag fields for the different instructions of unmodified object code in the instruction register.

12. The combination of claim 1 wherein:

the computer system has a particular instruction processing configuration;

and each instruction analyzer mechanism includes logic circuitry for implementing rules which define which types of instructions are compatible for parallel execution in the particular instruction processing configuration used for the computer system, such logic circuitry producing the compoundability signal for that analyzer mechanism.

13. The digital computer system as in claim 1, wherein means coupling said input means of said instruction compounding mechanism to an output of said larger-capacity, lower-speed storage mechanism for transferring instructions from said larger-capacity, lower-speed storage mechanism to said instruction compounding mechanism transfers said instructions in unmodified object code.

14. The digital computer system as in claim 1 wherein said data bits indicating parallel execution of said instruction pairs comprise a binary data bit associated with each instruction, said binary data bit indicating in one of its states that an associated instruction is a first instruction of a compoundable pair of instructions and said binary data bit indicating in its other state that an associated instruction is a second instruction of a compoundable pair of instructions, a pair of instructions indicated as compoundable only when a binary data bit associated with a first instruction is in said one state and a binary data bit associated with a second instruction is in said other state.

* * * * *

	Document ID	U	Title	Current OR
19	US 6609191 B1	☐	Method and apparatus for speculative microinstruction pairing	712/216
20	US 6578059 B1	☐	Methods and apparatus for controlling exponent range in floating-point calculations	708/496
21	US 6564315 B1	☐	Scheduler which discovers non-speculative nature of an instruction after issuing and reissues the instruction	712/214
22	US 6542984 B1	☐	Scheduler capable of issuing and reissuing dependency chains	712/214
23	US 6535973 B1	☐	Method and system for speculatively issuing instructions	712/218
24	US 6370639 B1	☐	Processor architecture having two or more floating-point status fields	712/222
25	US 6311261 B1	☐	Apparatus and method for improving superscalar processors	712/23
26	US 6212539 B1	☐	Methods and apparatus for handling and storing bi-endian words in a floating-point processor	708/495
27	US 6175910 B1	☐	Speculative instructions exection in VLIW processors	712/217
28	US 5951669 A	☐	Methods and apparatus for efficient control of floating-point status register	712/222
29	US 5911119 A	☐	Distributed instruction queue	712/214
30	US 5809267 A	☐	Apparatus and method for fast unified interrupt recovery and branch recovery in processors supporting out-of-order execution	712/218
31	US 5760776 A	☐	Branch operation system where instructions are queued until preparations is ascertained to be completed and branch distance is considered as an execution condition	712/233
32	US 5738457 A	☐	Scan chains for out-of-order load/store execution control	712/216
33	US 5732234 A	☐	Method and system for interrupt handling in a multi-processor computer system executing speculative instruction threads	712/23
34	US 5732244 A	☐	Multiple issue static speculative instruction scheduling with path tag and precise interrupt handling	712/23
35	US 5733336 A	☐	Method and apparatus for packing and unpacking wide instruction word using pointers and masks to shift word syllables to designated execution units groups	712/24
36	US 5730630 A	☐	Structure and method for instruction boundary machine state restoration	712/244
37	US 5580648 A	☐	Architectural support for software pipelining of nested loops	712/241
38	US 5926646 A	☐	Context-dependent memory-mapped registers for transparent expansion of a register file	712/32
39	US 5923871 A	☐	Multifunctional execution unit having independently operable adder and multiplier	712/221
40	US 5919256 A	☐	Operand cache addressed by the instruction address for reducing latency of read instruction	712/218
41	US 5915117 A	☐	Computer architecture for the deferral of exceptions on speculative instructions	710/262

UNITED STATES PATENT AND TRADEMARK OFFICE

CERTIFICATE OF CORRECTION

PATENT NO. : 5,475,853

DATED : December 12, 1995

INVENTOR(S) : Bartholomew Blaner et al.

It is certified that error appears in the above-identified patent and that said Letters Patent is hereby corrected as shown below:

Col 1 Line 16	**Delete "07/579,384" and insert --07/519,384 --**
Col 14 Line 62 claim 7	**Delete "bite" and insert --bits--**
Col 15 Line 3 claim 8	**Delete "Of" and insert --of--**

Signed and Sealed this

Eleventh Day of May, 1999

Attest:

Q. TODD DICKINSON

Attesting Officer

Acting Commissioner of Patents and Trademarks

	Document ID	U	Title	Current OR
42	US 5913048 A	☐	Dispatching instructions in a processor supporting out-of-order execution	712/215
43	US 5907860 A	☐	System and method of retiring store data from a write buffer	711/117
44	US 5889985 A	☐	Array prefetch apparatus and method	712/225
45	US 5887161 A	☐	Issuing instructions in a processor supporting out-of-order execution	712/244
46	US 5838940 A	☐	Method and apparatus for rotating active instructions in a parallel data processor	712/216
47	US 5835747 A	☐	Hierarchical scan logic for out-of-order load/store execution control	712/216
48	US 5809268 A	☐	Method and system for tracking resource allocation within a processor	712/200
49	US 5799179 A	☐	Handling of exceptions in speculative instructions	712/234
50	US 5794029 A	☐	Architectural support for execution control of prologue and eplogue periods of loops in a VLIW processor	712/241
51	US 5754812 A	☐	Out-of-order load/store execution control	712/216
52	US 5751985 A	☐	Processor structure and method for tracking instruction status to maintain precise state	712/218
53	US 5719800 A	☐	Performance throttling to reduce IC power consumption	713/321
54	US 5673426 A	☐	Processor structure and method for tracking floating-point exceptions	712/244
55	US 5673408 A	☐	Processor structure and method for renamable trap-stack	712/216
56	US 5659721 A	☐	Processor structure and method for checkpointing instructions to maintain precise state	712/228
57	US 5651515 A	☐	Processor structure and method for watchpoint of plural simultaneous unresolved branch evaluation	712/239
58	US 5651124 A	☐	Processor structure and method for aggressively scheduling long latency instructions including load/store instructions while maintaining precise state	712/215
59	US 5649136 A	☐	Processor structure and method for maintaining and restoring precise state at any instruction boundary	712/244
60	US 5644742 A	☐	Processor structure and method for a time-out checkpoint	712/244
61	US 5611063 A	☐	Method for executing speculative load instructions in high-performance processors	712/205
62	US 5584009 A	☐	System and method of retiring store data from a write buffer	711/117

HPS Trailer Page
for

Walk-Up_Printing

UserID: f

Printer: cpk2_2c21_gbkbptr

Summary

Document	Pages	Printed	Missed	Copies
US005701430	17	17	0	1
US005459844	19	19	0	1
US005475853	17	17	0	1
Total (3)	53	53	0	-

	Document ID	U	Title	Current OR
1	JP 2003208251 A	☐	ELECTRONIC PROCEDURE EXECUTING PROGRAM AND ELECTRONIC PROCEDURE EXECUTING SERVER	
2	JP 2003092752 A	☐	MULTIPLEXER, MULTIPLEXING METHOD, IMAGE DECODER AND IMAGE DECODING METHOD	
3	JP 2003039701 A	☐	PRINTER AND ITS NONPRINT MICROVIBRATION CONTROLLING METHOD	
4	JP 2001306127 A	☐	JOB INSTRUCTION METHOD	
5	JP 2001109597 A	☐	NETWORK PRINTING SYSTEM	
6	JP 2000156868 A	☐	METHOD AND DEVICE FOR DECODING IMAGE	
7	JP 11296389 A	☐	CONTEXT CONTROLLER FOR MANAGING MULTI-TASKING BY PROCESSOR	
8	JP 11272383 A	☐	METHOD AND DEVICE FOR GENERATING ACTION SYNCHRONIZED TYPE VOICE LANGUAGE EXPRESSION AND STORAGE MEDIUM STORING ACTION SYNCHRONIZED TYPE VOICE LANGUAGE EXPRESSION GENERATING PROGRAM	
9	JP 11247689 A	☐	AIR-FUEL RATIO CONTROL DEVICE OF INTERNAL COMBUSTION ENGINE	
10	JP 10266886 A	☐	FUEL CUT CONTROL DEVICE OF INTERNAL COMBUSTION ENGINE	
11	JP 09223013 A	☐	SPECULATIVE LOAD INSTRUCTION EXECUTING METHOD FOR HIGH PERFORMANCE PROCESSOR	
12	JP 08298615 A	☐	ELECTRONIC EQUIPMENT	
13	JP 08171645 A	☐	METHOD FOR PRODUCING PICTURE AND ITS DEVICE	
14	JP 08171048 A	☐	CAMERA	
15	JP 08102365 A	☐	LIGHTING CONTROL SYSTEM	
16	JP 08072324 A	☐	TAPELIKE LABEL FORMING APPARATUS	
17	JP 08036407 A	☐	PROGRAMMABLE CONTROLLER AND ITS SFC PROGRAM EXECUTION METHOD	
18	JP 07257756 A	☐	CONTROL METHOD FOR PICKING OUT POWDERY-GRANULAR MATERIAL	
19	JP 07152726 A	☐	MODIFIED CHOLESKY DECOMPOSITION CALCULATING DEVICE	
20	JP 06301099 A	☐	INFORMATION RECORDER FOR CAMERA	

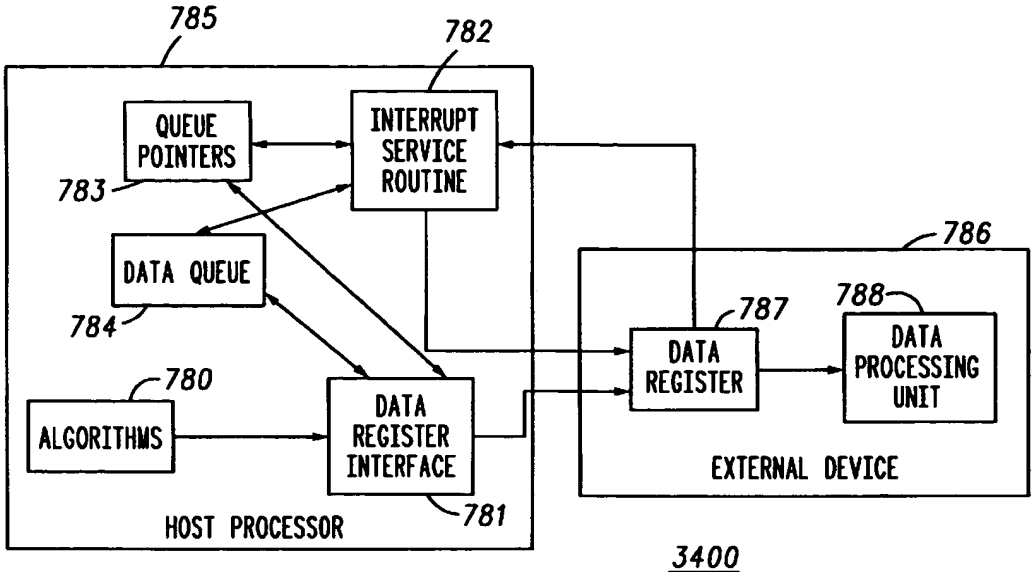
785
782
QUEUE POINTERS
INTERRUPT SERVICE ROUTINE
783
DATA QUEUE
784
786
787
788
DATA REGISTER
DATA PROCESSING UNIT
780
ALGORITHMS
DATA REGISTER INTERFACE
781
EXTERNAL DEVICE
HOST PROCESSOR
3400
— PRIOR ART —

FIG.34

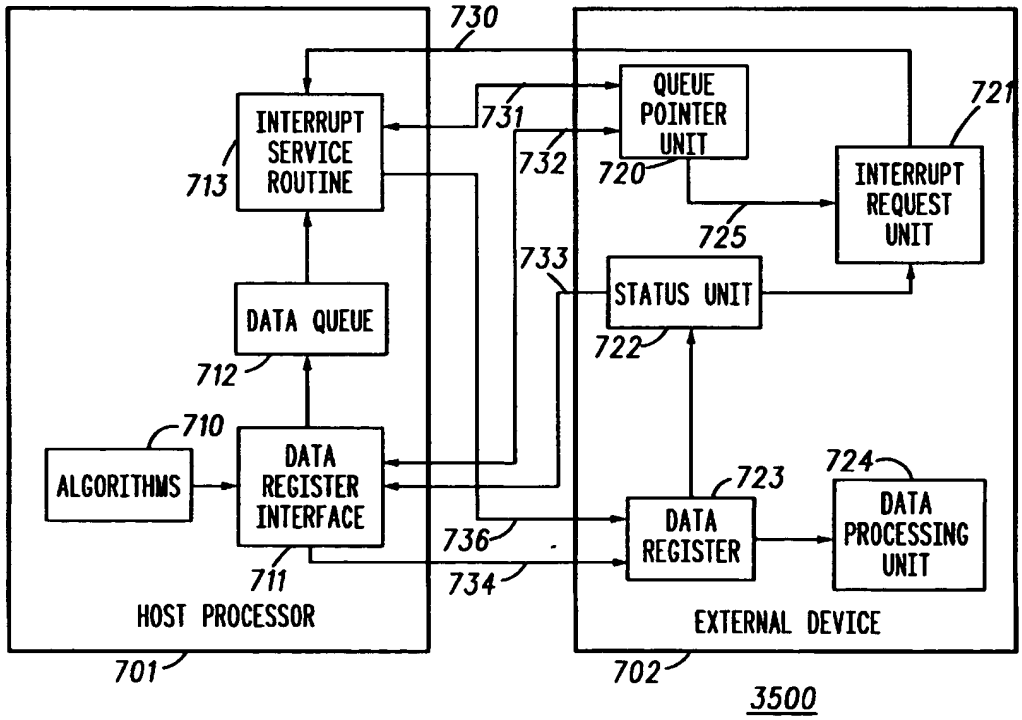
730
731
732
QUEUE POINTER UNIT
720
721
INTERRUPT REQUEST UNIT
725
713
INTERRUPT SERVICE ROUTINE
733
STATUS UNIT
722
DATA QUEUE
712
710
ALGORITHMS
DATA REGISTER INTERFACE
723
724
736
734
DATA REGISTER
DATA PROCESSING UNIT
711
HOST PROCESSOR
EXTERNAL DEVICE
701
702
3500

FIG.35

	Document ID	U	Title	Current OR
21	JP 06201 802 A	☐	INTEGRATED CIRCUIT WITH TESTING FUNCTION	
22	JP 06160 042 A	☐	INSTRUCTION METHOD FOR MOUNTED PART INSPECTION DEVICE	
23	JP 06028 198 A	☐	RISC MICROPROCESSOR PRIORITY VECTOR INTERRUPT SYSTEM	
24	JP 05233 276 A	☐	METHOD AND DEVICE FOR SUBMODEL CONTROL IN DATA PROCESSING SYSTEM	
25	JP 05143 858 A	☐	POS TERMINAL	
26	JP 05088 892 A	☐	METHOD AND DEVICE FOR PARALLELLY PROCESSING INFORMATION	
27	JP 05053 836 A	☐	METHOD FOR AUTOMATICALLY DECIDING TASK EXECUTION PRIORITY ORDER	
28	JP 04352 082 A	☐	IMAGE PROCESSOR	
29	JP 04280 263 A	☐	IMAGE FORMING DEVICE	
30	JP 04170 126 A	☐	LINE CONTROL ADAPTOR	
31	JP 03285 569 A	☐	CONTROLLER OF POWER INVERTER	
32	JP 03273 284 A	☐	SCANNING METHOD FOR SCANNING TYPE DISPLAY	
33	JP 03024 560 A	☐	ELECTROPHOTOGRAPHIC DRY TONER	
34	JP 01154 236 A	☐	EXECUTING DEVICE FOR TIME-DIVISION TASK	
35	JP 01135 650 A	☐	DUPLICATE PRINTING INHIBITION METHOD	
36	JP 01133 107 A	☐	SEQUENCE CONTROLLER	
37	JP 01092 843 A	☐	DATA PROCESSING UNIT	
38	JP 01076 486 A	☐	MEMORY IC	
39	JP 01042 751 A	☐	MEMORY CONTROL SYSTEM	
40	JP 63191 346 A	☐	BRAKE MECHANISM	
41	JP 63104 535 A	☐	COMMUNICATION CONTROL METHOD FOR DATA TERMINAL EQUIPMENT	
42	JP 63047 838 A	☐	KNOWLEDGE SUCCESSION CONTROL SYSTEM	
43	JP 62272 314 A	☐	MICROCOMPUTER	

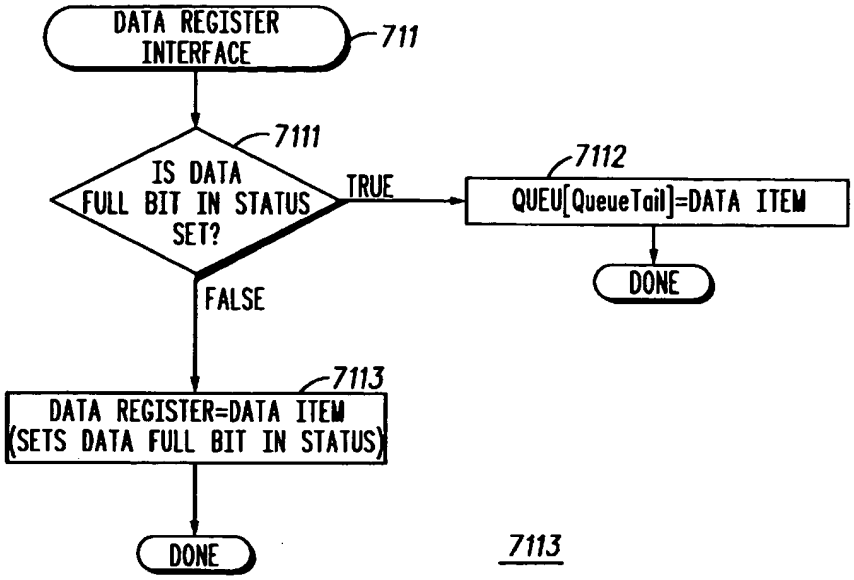
DATA REGISTER INTERFACE
711
7111
IS DATA FULL BIT IN STATUS SET?
TRUE
7112
QUEU[QueueTail]=DATA ITEM
DONE
FALSE
7113
DATA REGISTER=DATA ITEM (SETS DATA FULL BIT IN STATUS)
DONE
7113

FIG. 37

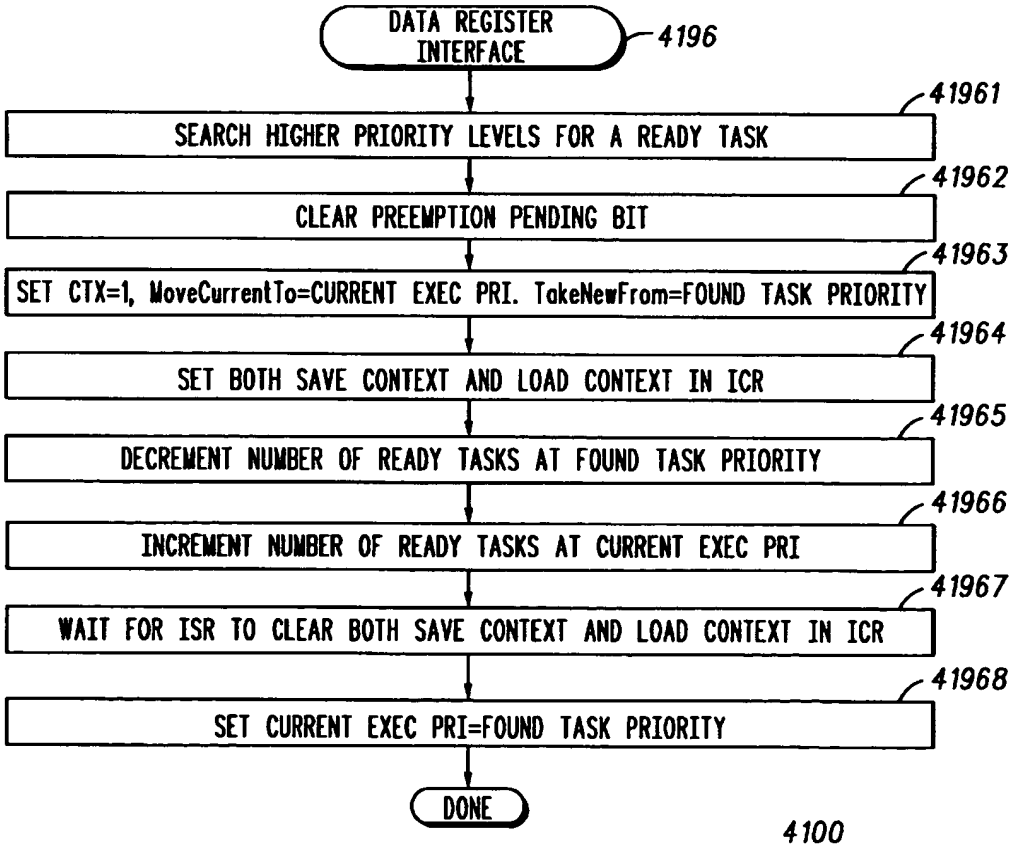
DATA REGISTER INTERFACE
4196
41961
SEARCH HIGHER PRIORITY LEVELS FOR A READY TASK
41962
CLEAR PREEMPTION PENDING BIT
41963
SET CTX=1, MoveCurrentTo=CURRENT EXEC PRI. TakeNewFrom=FOUND TASK PRIORITY
41964
SET BOTH SAVE CONTEXT AND LOAD CONTEXT IN ICR
41965
DECREMENT NUMBER OF READY TASKS AT FOUND TASK PRIORITY
41966
INCREMENT NUMBER OF READY TASKS AT CURRENT EXEC PRI
41967
WAIT FOR ISR TO CLEAR BOTH SAVE CONTEXT AND LOAD CONTEXT IN ICR
41968
SET CURRENT EXEC PRI=FOUND TASK PRIORITY
DONE
4100

FIG. 41

	Document ID	U	Title	Current OR
44	JP 62197 819 A	☐	DATA TRANSFER CONTROL SYSTEM	
45	JP 62197 818 A	☐	DATA TRANSFER CONTROL SYSTEM	
46	JP 62173 100 A	☐	OPERATING METHOD FOR HYDRAULIC PRESS	
47	JP 62072 045 A	☐	STORAGE PROTECTION KEY CONTROLLER	
48	JP 61099 996 A	☐	DYNAMIC RAM CONTROLLING SYSTEM	
49	JP 60020 211 A	☐	SEQUENCE CONTROLLING DEVICE	
50	JP 59223 463 A	☐	RECORDING DEVICE	
51	JP 59060 645 A	☐	SYSTEM FOR PROCESSING INSTRUCTION EXECUTION OF PROCESSOR	
52	JP 58224 807 A	☐	METHOD FOR CONTROLLING AIR CONDITIONER FOR AUTOMOBILE	
53	JP 58156 250 A	☐	METHOD FOR REPLACING CONTROLLER OF DATA COMMUNICATION NETWORK	
54	JP 58115 552 A	☐	INTERRUPT CONTROLLING SYSTEM	
55	JP 56096 367 A	☐	INFORMATION PROCESSING DEVICE	
56	JP 55043 662 A	☐	ORDER RE-EXECUTION INFORMATION PROCESS SYSTEM	
57	JP 54150 051 A	☐	CHARACTERISTIC TEST DEVICE	
58	JP 54147 388 A	☐	SEQUENCE CONTROLLER	
59	JP 54147 386 A	☐	SEQUENCE CONTROLLER	
60	JP 53121 443 A	☐	INPUT UNIT OF FRONT BUSINESS PROCESSING SYSTEM	
61	EP 13311 39 A2	☐	Information providing apparatus, and method	
62	EP 13108 64 A2	☐	Method and circuit for conditional-flag rewriting control	
63	WO 30072 72 A1	☐	SYSTEMS AND METHODS FOR INTERACTIVE TRAINING OF PROCEDURES	
64	GB 22849 12 A	☐	Data processor with speculative instruction fetching and method of operation	
65	EP 55271 7 A2	☐	Data processing system and method using virtual storage system.	
66	EP 53699 1 A1	☐	Method and apparatus for resynchronizing a moving rotor of a polyphase DC motor.	

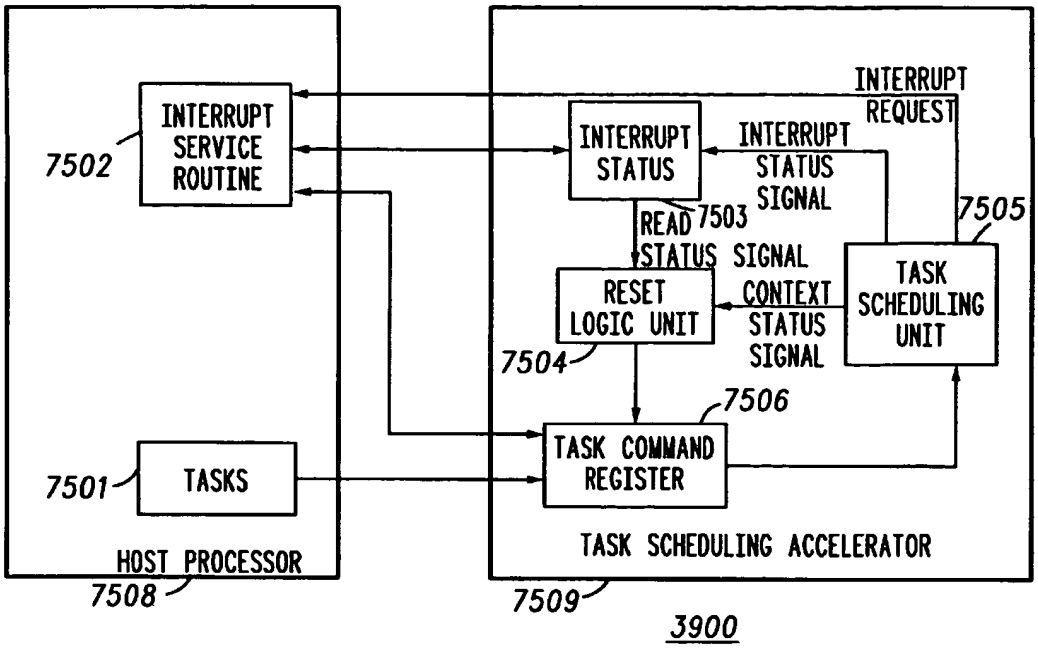
INTERRUPT SERVICE ROUTINE
7502
INTERRUPT STATUS
INTERRUPT REQUEST
INTERRUPT STATUS SIGNAL
READ STATUS SIGNAL
7503
7505
RESET LOGIC UNIT
CONTEXT STATUS SIGNAL
TASK SCHEDULING UNIT
7504
7506
TASKS
7501
TASK COMMAND REGISTER
HOST PROCESSOR
7508
TASK SCHEDULING ACCELERATOR
7509
3900

FIG. 39

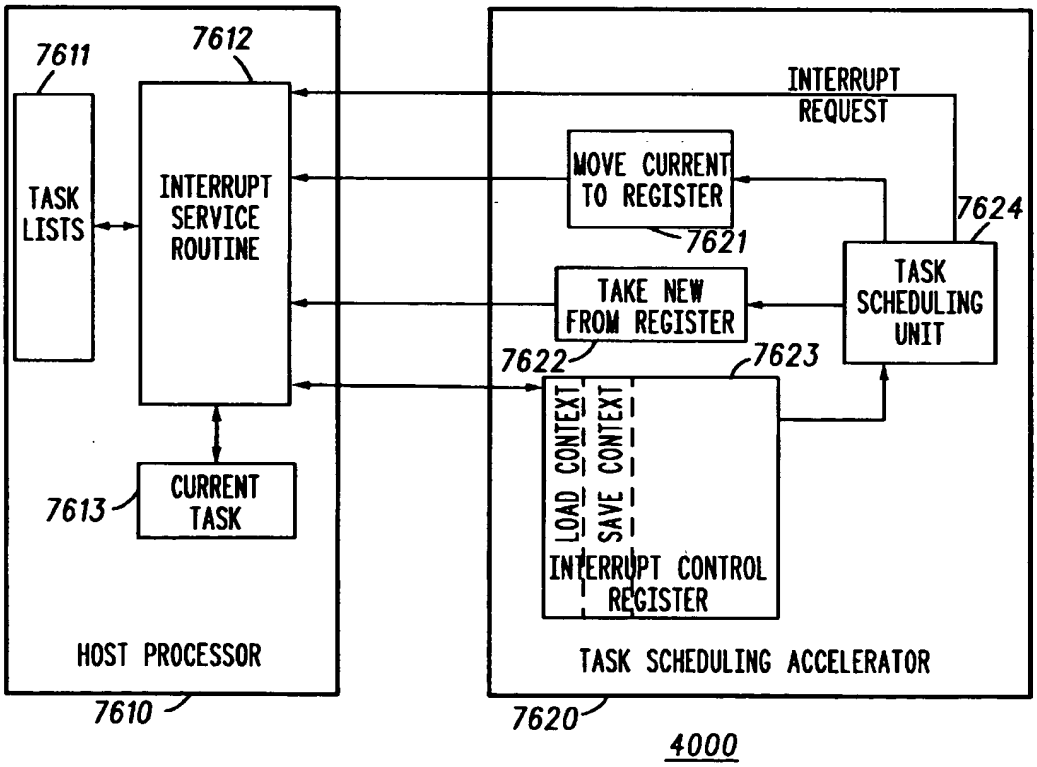
7611
7612
TASK LISTS
INTERRUPT SERVICE ROUTINE
INTERRUPT REQUEST
MOVE CURRENT TO REGISTER
7621
7624
TASK SCHEDULING UNIT
TAKE NEW FROM REGISTER
7622
7623
LOAD CONTEXT
SAVE CONTEXT
INTERRUPT CONTROL REGISTER
CURRENT TASK
7613
HOST PROCESSOR
7610
TASK SCHEDULING ACCELERATOR
7620
4000

FIG. 40

	Document ID	U	Title	Current OR
67	EP 438961 A2	☐	Hardware data string operation controller.	
68	WO 9014723 A1	☐	BIT STREAM MACHINE	
69	EP 376004 A2	☐	Variable length pipe operations sequencing.	
70	EP 328289 A2	☐	IC card and method of writing its operation program.	
71	EP 301707 A2	☐	Apparatus and method for providing an extended processing environment on nonmicrocoded data processing system.	
72	EP 254352 A1	☐	A programmable machine system.	
73	EP 63458 A2	☐	Microcomputer system.	
74	EP 35647 A2	☐	A SIMD data processing system.	
75	NN9612119	☐	Instruction Match Function for Processor Performance Monitoring	
76	US 20040049000 A	☐	Dynamic random access memory has local control circuit to generate local control signals based on rise and fall of received main control signals	
77	US 20030208665 A	☐	Cache hit/miss prediction method in pipelined processor, involves reading cache hit/miss prediction value associated with entry corresponding to memory address	
78	WO 2003091411 A	☐	Obtaining semi-synthetic protein-based site directed probe e.g., ubiquitin vinyl sulfone specific for deubiquitinating enzymes, useful for identification of subset of proteome e.g., ubiquitin/ubiquitin-like proteins	
79	US 20030187905 A	☐	Task sequence scheduling method in word processor, involves maintaining task list containing entry sequence of enabled and disabled tasks and scheduling successive block to be executed after previous block	
80	EP 1331539 A	☐	Operating method for digital system, by jumping to entry address, executing activation sequence, and entering secure mode.	
81	WO 2003046715 A	☐	Central processing device e.g. CPU, activates instruction holding unit to set instruction corresponding to selected operating mode in order to acquire firmware	
82	EP 1316868 A	☐	Electronic control system for an implement towing agricultural vehicle has devices enabling driver during recording mode to record brake signal in addition to recording of sequence of control steps being executed by driver	
83	EP 1310864 A	☐	Conditional flag rewriting control method for processor e.g. Reduced Instruction Set Computer determines if conditional flag rewriting is enabled or disabled then rewrites/does not rewrite flag as appropriate	
84	WO 2003032569 A	☐	Server initiated synchronization for third generation CDMA wireless communications that involves server determining maximum size of message to be sent to mobile station for session initiation request	
85	KR 2002096465 A	☐	Tempo controller of digital audio decoder chip	

DATA TRANSFER USING SOFTWARE INTERRUPT SERVICE ROUTINE BETWEEN HOST PROCESSOR AND EXTERNAL DEVICE WITH QUEUE OF HOST PROCESSOR AND HARDWARE QUEUE POINTERS ON EXTERNAL DEVICE

RELATED PATENT APPLICATIONS

The present patent application is related to the following patent applications which are being filed simultaneously (and are hereby incorporated herein) and are assigned to the same entity, Motorola, Inc.: U.S. patent application Ser. No. 09/037,173, still pending, Method, Device and Article of Manufacture for Efficient Task Scheduling in a Multi-tasking Preemptive Priority-Based Real-Time Operating System, by Brett Louis Lindsley; U.S. patent application Ser. No. 09/041,101, still pending, Method, Device and Article of Manufacture for Implementing a Real-Time Task Scheduling Accelerator, by Brett Louis Lindsley, Uri Dayan and Moshe Tarrab; U.S. patent application Ser. No. 09/037,599, still pending, Method, Device and Article of Manufacture for Specifying and Controlling a Task Switch from an External Device in a Multi-Tasking Preemptive Priority-Based Real-Time Operating System, by Brett Louis Lindsley and Ed Martinez; and U.S. patent application Ser. No. 09/041,189, still pending Method, Device and Article of Manufacture for Clearing Command Bits in a Task Scheduling Device for a Multi-Tasking Preemptive Priority-Based Real-Time Operating System, by Brett Louis Lindsley.

FIELD OF THE INVENTION

The field of the invention relates to multiple task execution processors, in particular, a multi-task execution processor having a preemptive prioritized task scheduling system.

BACKGROUND OF THE INVENTION

Multi-tasking is the capability of a single processor to execute multiple tasks concurrently. A processor performs multi-tasking execution by switching between tasks using a defined set of multi-tasking rules. For theoretical reasons, real-time multi-tasking systems preferably use a set of rules where tasks are prioritized. Lower priority tasks can be preempted (forcibly switched) to execute a higher priority task if necessary. This "preemptive prioritized" system always executes the highest priority ready task.

A "task" is defined as an independent entity of computing. For example, where a computer system permits many users on different terminals access to the same processor, the computer system assigns a unique task to each user's computing (where each user's computing is unrelated to another user's computing). Generally, a task unit contains an entry for each task. Tasks are typically synchronized using "events". An event is used to indicate that an activity has taken place, such as data arrival, time-out, etc. Thus, an event may indicate execution of a task, an interrupt service routine or the like. Events are counted using semaphores. Semaphores synchronize the event producer and the event consumer as shown in FIG. 2, numeral **200**.

Events are generated by tasks **201** or interrupt service routines (ISRs) **202**. ISRs are not allowed to consume events because ISRs are not scheduled and execute outside of the normal task scheduling environment of the processor. However, ISRs are allowed to generate events to indicate the status of the cause of the interrupt.

A counting semaphore **203** allows a plurality of events to be generated without being lost. Similarly, a plurality of tasks may be waiting for the same event. When the semaphore counter is positive, it indicates events have been produced but no tasks are waiting for the events. When the semaphore counter is zero, it indicates no events have been generated and no tasks are waiting. When the semaphore counter is negative, it indicates there are no events generated but a plurality of tasks are waiting for the events. The process of generating an event to a semaphore is typically called "posting" the event **204**. The process of waiting for an event from a semaphore is typically called "pending" for the event **205**.

When an event is generated for a semaphore that has a list of tasks pending, the task **206** at the head of the list becomes ready for execution. At this time, the task scheduling function **207** must determine if this task should be selected for execution or if it should be put in a list with other ready tasks to be executed at a later time. This decision is based on the priority of the task that is currently executing compared to the priority of the task that has just become ready.

If the task that has just become ready is the same or lower priority than the currently executing task, the currently executing task continues executing. The new ready task is moved to a list to be executed at a later time.

If the task that has just become ready is a higher priority than the currently executing task, the currently executing task is preempted to allow the new ready task to execute. Thus, the highest priority ready task **208** is always selected for execution.

At some time, the currently executing task completes its computation and becomes inactive. A task becomes inactive by pending on a semaphore when the semaphore count is zero or negative. For example, a semaphore may represent data buffers that require processing. If the task that processes data buffers pends for the semaphore that represents data buffers, the task is synchronized to the data buffer generation. If data buffers are available, the semaphore count is greater than zero. Task pend requests on the semaphore allow the task to continue. If data buffers are not available, the semaphore count is less than or equal to zero. When the task pends on the semaphore with a count that is less than or equal to zero, the task does not have data for processing and will block execution.

At the point when a task pends on a semaphore with zero or negative count, the task that has made the request is not allowed to continue execution and must "block". The task scheduling function stops this task and selects another task for execution. The task scheduling function looks at other ready tasks at the same or lower priorities to determine the next highest priority ready task. If a task is found, it is selected for execution. If a task is not found, the processor is put into an idle state until a task becomes ready.

Although the task selection process is easy to understand, in a practical environment this can lead to significant inefficiencies. For example, the following cause inefficiencies in a preemptive prioritized task scheduling system:

- Each time a semaphore is processed, there is a potential for a task change. This causes the scheduling function to be called quite often when many times there is no change.
- When a task pends for a semaphore and blocks, the scheduler must perform a search algorithm to determine the next ready task. If there are a large number of tasks and priority levels, this activity may be time-consuming.
- The process of searching for a task may have variable timing making task scheduling functions non-deterministic.

	Document ID	U	Title	Current OR
86	US 20020 15236 8 A	☐	Processor for computer system, executes subsequent instruction according to prediction of value indicating result of execution of instruction	
87	JP 20021 23596 A	☐	Car inspection and registration method using internet, involves performing package management of car owner and mechanic field to enable car inspection agent to execute car inspection and registration process	
88	EP 11687 72 A	☐	Controlling operation of WAP enabled mobile terminal in telecommunication network, incorporating preferential mode selection instruction in sequence of instructions of page of service data	
89	JP 20012 23938 A	☐	Camera with built-in video recording device, has selection unit to choose automatic execution mode to operate device with respect to stored program	
90	US 62302 78 B	☐	Data processing device for audio reproducing system, has processing units with respective instruction sequencers to process sequence of instructions selected from respective memory on receiving instruction directive	
91	US 62124 08 B	☐	Voice controlled interaction enabling for voice mail system, by interpreting audible information from user in voice command mode as voice command instructing communication handset to dial needed key sequence entry	
92	JP 20003 11931 A	☐	Semiconductor integrated circuit for large-scale integration, has test sequencer which activates CPU to execute stored test program of intellectual core property, corresponding to setup mode value in register	
93	GB 23459 80 A	☐	Mode shape converter arranged at terminal of optical device and adapted to couple light via optical fibers, has lower cladding coated over substrate having etched portion with lower rib waveguide over which core is formed	
94	US 60723 95 A	☐	Remote controlled signaling method for use in classrooms, involves activating red, yellow and green lights sequentially, to indicate students about the termination of instruction period	
95	WO 20001 6570 A	☐	Electrostatic radio frequency identification reader/encoder, has exciter that generates and transmits data sequence signal with data sequence received from processor, and clock signal	
96	US 59833 35 A	☐	Multiple out-of-order instructions issuing and execution mechanism in superscalar machine e.g. IBM system	
97	US 59580 45 A	☐	Memory operands specialized fetching method in load store unit for code sequence execution in microprocessor	
98	US 61340 63 A	☐	Multitrack data transfers using disk drive storage device, such as computer disk drive, minimizing microprocessor firmware overhead and reducing amount of track skew	
99	US 58954 92 A	☐	Data integrity protecting method in computer system - involves setting lock field of current lock entry to lock state, if no other lock entry is found to have lock field set in locked state	
100	US 58729 10 A	☐	Parity error injection system for instruction processor in data processing system	
101	JP 10149 295 A	☐	Sequence controller for computer system connected with several copiers - has instruction decoder which decodes from instruction read-out and sets condition execution instruction flag active to read condition execution instruction from memory	
102	US 57404 17 A	☐	Low power pipelined processor - includes BHT storing prediction states receiving branch instruction address and providing entry history bits with pipeline resource switching to low power mode when enable signal received	

The task scheduler must obtain control of the processor to switch tasks.

In a software task scheduling system, interrupt service routines typically require entry and exit code to flag interrupt usage and nesting. The flagging is typically used to call the scheduler to determine if the interrupt generated an event that may have changed execution. The use of entry and exit code consumes time and is undesirable when processing interrupts.

Solutions to the problems set forth above typically involve constraining the functionality of the scheduler. These constraints may include limiting the number of tasks, limiting the number of priority levels, limiting the number of tasks at each priority level, limiting the representation range of the semaphore counter, etc. It is undesirable to have such constraints in a system because such constraints may restrict the type of system that can be supported.

Thus, there is a need for a method, device and article of manufacture that optimizes real-time multi-tasking for a processor.

SUMMARY

A combined hardware and software data queuing mechanism for efficient data transfer from a host processor to an external device is presented. In one embodiment, the host processor maintains a queue of data items and the external device maintains hardware pointers to the queue. Data is efficiently stored in the host processor's queue by reading the external hardware pointers when the external device data registers are full. Data is efficiently retrieved from the queue and written to the external device by an interrupt service routine by reading the external hardware pointers. The hardware queue pointers automatically increment and wrap when reaching a maximum value. Placing the queue pointers in hardware relieves the software of maintaining the pointers, incrementing them and wrapping them. Since the external hardware device has direct access to the queue pointers, the external hardware device asserts its interrupt to request data from the host processor only when it knows there is data queued in the host processor.

In one embodiment, a device provides efficient transfer of data items between a host processor and an external device. The device includes the host processor and the external device. The host processor is coupled to external queue pointers. The host processor is capable of queuing a plurality of data items and transferring the plurality of data items to the external device using an interrupt service routine unit. The external device is coupled to the host processor for storing and incrementing/decrementing the external queue pointers and processing the data items.

In another embodiment, a method for efficient transfer of data items between a host processor and an external device is provided. The method includes using a host processor which is coupled to external queue pointers for queuing a plurality of data items. The method further includes transferring the plurality of data items to the external device using an interrupt service routine unit, and using the external device which is coupled to the host processor for storing and manipulating the external queue pointers and processing the data items.

In another embodiment, an article of manufacture having a device for efficient transfer of data items between a host processor and an external device is provided. The device includes the host processor and the external device. The host processor is coupled to external queue pointers for queuing a plurality of data items and transferring the plurality of data items to the external device using an interrupt service routine unit. The external device is coupled to the host processor for storing and manipulating the external queue pointers, and processing the data items.

The foregoing is a summary and thus contains, by its very nature and by necessity, simplifications, generalizations and omissions of detail. Consequently, the reader will appreciate that the summary is illustrative only and is not intended to be limiting in any way. Other aspects, inventive features, and advantages of the present invention, as defined solely by the claims, will become apparent to one of ordinary skill in the art based on the teachings of the non-limiting detailed description set forth below.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1 is a block diagram of one embodiment of a device in accordance with the present invention, showing data flow and functionality of the invention.

FIG. 2 is a diagrammatic representation of event processing and schedule processing with respect to the basic operation of an event driven preemptive prioritized task multi-tasking system in accordance with the present invention.

FIG. 3 shows schematic representations of examples of special function bits in control registers for task commands, interrupt control and status in accordance with the present invention describes.

FIG. 4 is a flow chart showing one embodiment of steps for implementing the operation of the state machine that controls the accelerator in accordance with the present invention.

FIG. 5 is a flow chart showing one embodiment of data flows for executing commands that are synchronous to the task wherein the task must allow the previous command to clear before executing the next command in accordance with the present invention.

FIG. 6 is a flow chart indicating one embodiment of the functions that tasks and interrupts use to control the task scheduling accelerator in accordance with the present invention.

FIG. 7 is a flow chart showing one embodiment of steps for adding a new task to the currently executing task set in the multi-tasking environment in accordance with the present invention.

FIG. 8 is a flow chart illustrating one embodiment of steps for how a task exits to remove itself from the currently executing task set in the multi-tasking environment in accordance with the present invention.

FIG. 9 is a flow chart showing one embodiment of steps for how a task may modify its own priority in accordance with the present invention.

FIG. 10 is a flow chart showing one embodiment of steps for how a task may yield to another task at the same priority level in accordance with the present invention.

FIG. 11 is a flow chart showing one embodiment of steps for how tasks pend for semaphores on the host processor in accordance with the present invention.

FIG. 12 is a flow chart showing one embodiment of data flows for executing commands that are asynchronous to the task (commands that the task does not need to wait for completion or commands generated by interrupts) in accordance with the present invention.

FIG. 13 is a flow chart showing one embodiment of steps for how semaphores are posted by tasks and interrupts on the host processor in accordance with the present invention.

	Document ID	U	Title	Current OR
103	RD 408186 A	☐	Testing method for program activity implementations of work-flow management systems e.g. IBM FlowMark (RTM) - constructing network of activities as coloured graph, with nodes representing activities to be performed and edges potential sequence of their execution, and materialising input container, requesting fields	
104	GB 2310057 A	☐	Pipeline type information processing apparatus - has instruction read inhibit bit provided for each pair of addresses registered in branch history and instruction read inhibit bit setting section	
105	US 5611063 A	☐	Selective execution method for CPU speculative load instructions - involves determining bit state in response to load instruction, suspending instruction execution pending branch processor control signals and resetting bit state on execution of instruction	
106	US 5644742 A	☐	Speculative instruction execution tracking for processor - involves assigning identification tag to each instruction issued with associated activity bit set on issue and cleared on completion	
107	CN 1127912 A	☐	Speech synthesising device for high level language command decoding	
108	US 5506821 A	☐	Optical disk program repeater system - has sequence controller sending predetermined set of instructions to disk player in response to power supply, and disk output monitor controlling disc player in response to proprietary code recorded on disk	
109	US 5490254 A	☐	Integrated circuit for interfacing system with MIL-STD-1533 data bus - uses register-based architecture that allows for autonomous operation in three standard modes simultaneously, with operation of IC being autonomous	
110	US 5483539 A	☐	High frequency programmable demultiplexer appts. for PCM-TDM data communication - includes FAL hardware controlled by PFAC with decommutation and dejustification function unit with output logic controlled by programmable decommutator sequence	
111	US 5428802 A	☐	Disc data storage retrieval system controller - has device for executing critical selection start command received from host computer which directs controller flag in on state	
112	US 5416911 A	☐	Instruction pipeline data processor with load multiple register instruction - stores identity of highest numbered register or registers modified during last execution cycle of multiple register loading instruction with indication of number of registers loaded	
113	EP 614193 A	☐	Integrated circuit memory retention fault detection - causing each memory bank to execute three test sequences, each sequence delayed to allow detection or bit pattern from previous sequence	
114	EP 677188 B	☐	Instruction tagging for superscaler processor - has tags in FIFO allocated to instructions fetched into execution units and used to monitor program order of instructions and registers used	
115	US 5463743 A	☐	Actively patching SCSI processor instructions for reselection operation with tagged queues - preparing jump table comprising jump to sequence for each possible tagged queue, obtaining tag value of reselecting queue, preparing jump into jump table having offset from beginning of jump table based on tag value and executing jump	
116	EP 581659 A	☐	Computerised navigation system for aircraft - has LAN with central unit controlling access across network and to memory which stores navigation information retrieved and processed by suite of programs while terminals display results	
117	EP 578540 A	☐	Function testing of Application Specific Integrated Circuit - using silicon chip with application program connected to central unit with read-only memory to test at time of production	
118	JP 05035442 A	☐	Data processing appts. with video interface function for serially transferring image data to printer, display - inputs 32-bit image data to thinning out circuit through bus and thins out data to have active data width of 24 bits	

FIG. **14** is a schematic representation of operation of pointer registers for assisting the queuing of semaphore posts in accordance with the present invention.

FIG. **15** is a flow chart showing one embodiment of steps for how the state machine requests semaphore posts to be moved from the queue on the host processor to the task scheduling accelerator in accordance with the present invention.

FIG. **16** is a schematic representation of the data flows and information for the accelerator to signal the host processor and communicate changes in the task execution in accordance with the present invention.

FIG. **17** is a flow chart showing one embodiment of steps for the functions executed by the interrupt service routine on the host processor in accordance with the present invention.

FIG. **18** is a flow chart showing one embodiment of steps for how the interrupt service routine moves a queued semaphore post to the accelerator in accordance with the present invention.

FIG. **19** is a schematic representation of data structures stored on the host processor used to describe the current state of task execution in accordance with the present invention.

FIG. **20** is a schematic representation of how the interrupt interface communicates information that determines how the interrupt service routine switches tasks in accordance with the present invention.

FIG. **21** is a flow chart showing one embodiment of steps for how the interrupt service routine switches tasks in accordance with the present invention.

FIG. **22** is a schematic representation of how processing on the host processor and task scheduling accelerator overlap to provide efficient task switching in accordance with the present invention.

FIG. **23** is a flow chart showing one embodiment of steps for how the state machine processes adding task requests from the host processor in accordance with the present invention.

FIG. **24** is a flow chart showing one embodiment of steps for how the state machine processes task exit requests from the host processor in accordance with the present invention.

FIG. **25** is a flow chart showing one embodiment of steps for how the state machine processes modify priority requests from the host processor in accordance with the present invention.

FIG. **26** is a flow chart showing one embodiment of steps for how the state machine processes task yield requests from the host processor in accordance with the present invention.

FIG. **27** is a flow chart showing one embodiment of steps for how the state machine processes semaphore pend requests from the host processor in accordance with the present invention.

FIG. **28** is a flow chart showing one embodiment of steps for how the state machine processes semaphore post requests from the host processor in accordance with the present invention.

FIG. **29** is a flow chart showing one embodiment of steps for how the interrupt service routine moves a task between queues in accordance with the present invention.

FIG. **30** is a block diagram illustrating how the task scheduling accelerator is physically connected to a host processor in accordance with the present invention.

FIG. **31** is a block diagram schematic showing one embodiment of how the task scheduling accelerator may be implemented in a hardware configuration in accordance with the present invention.

FIG. **32** is a block diagram schematic showing one embodiment of the state machine that controls the hardware in accordance with the present invention.

FIG. **33** is a schematic representation of examples of encoding of the instructions for the state machine in accordance with the present invention.

FIG. **34** is a block diagram illustrating how data is typically transferred between a software task or interrupt on a host processor to an external device in the prior art.

FIG. **35** is a block diagram illustrating the specific details of implementing an efficient data queuing method.

FIG. **36** is a block diagram illustrating the specific details of how hardware queue pointers operate.

FIG. **37** is a flow chart illustrating the required operations by an interface function on the host processor for either writing data to the external device or queuing the data.

FIG. **38** is a flow chart illustrating how the interrupt service routine efficiently moves data from the host processor to the external device.

FIG. **39** is a block diagram illustrating specific details of how the task command register is cleared.

FIG. **40** is a block diagram illustrating specific details of how the Task Scheduling Accelerator efficiently communicates information about a task switch to the Host Processor.

FIG. **41** is a flow chart illustrating how the Task Scheduling Accelerator selects a task after the task lock has been turned off and higher priority ready tasks are pending.

DETAILED DESCRIPTION OF A PREFERRED EMBODIMENT

The present invention resolves the problems with implementing task scheduling without the need for imposing constraints by moving a selected portion of the task scheduling functionality to an external hardware accelerator. By allowing part of the functionality to remain in software on the host processor, flexibility is maintained without the need of constraints. By moving scheduling decisions to an external accelerator device, fast determination of scheduling decisions is performed. Since the external accelerator device may compute schedule decisions in parallel with activity on the host processor, an overlap of scheduling decisions with host processor activity allows variable or lengthy scheduling decisions to be hidden during host processor activity.

The present invention improves upon software task scheduling implementations because software implementations are typically slow, require interrupt service routine entry and exit functions, and execute scheduling functions in serial with task execution.

The present invention also improves upon prior art hardware devices. Prior art hardware devices store almost all of the information for task scheduling locally in the hardware. To minimize the amount of information stored (and reduce the hardware cost), prior art devices require restrictions to limit the number of tasks, the number of tasks at the same priority level, the range of the semaphore counter, etc.

The present invention improves upon the prior art by providing a method of determining a ready task that is dependent only on the number of priority levels and not the number of tasks. This allows many tasks to be added to the system without an increase in the search time for a ready task. Further, scheduling computations for determining ready tasks is overlapped with the host processor for maximum efficiency. Because scheduling computations are performed externally to the host processor, no entry/exit code is required in interrupt service routines. The present invention

	Document ID	U	Title	Current OR
119	US 5159676 A	☐	DRAM controller IC providing pseudo-cache mode operation using standard page mode draws - uses logic to enable or disable page mode operation as result of executing software instructions, or prediction of page mode efficiency based on past performance	
120	US 5163140 A	☐	Branch prediction cache structure responds to input program counter - using two levels that respond to both small and large entry numbers	
121	SU 1665379 A	☐	Programmed with operative correction execution sequence monitor - includes decoder with outputs connected to corresp. inputs of delay elements	
122	EP 430453 A	☐	Error recovery in optical disk cartridge handling system - stopping low level moving function state update flag is set, and recovery is performed after completion of high level moving	
123	EP 423726 A	☐	Branch control circuit for pipeline processing of instruction - predicts whether branch is taken and processes instructions addressed by branch before it is taken	
124	EP 410516 A	☐	Coding technique for binary data - uses data octets to impose binary encoding to protect data using masking system and correspondence table	
125	EP 407295 A	☐	Monitoring of execution of program in computer system - uses access to memory and filtering of I=O transactions to provide selected operational data to external programs	
126	GB 2232281 A	☐	IC card memory protection and test program - protects predetermined memory region including stored pass code and address restriction circuit on address bus	
127	EP 397934 A	☐	Bit stream machine - has orchestrator with program memory, bit controller, LFS sequencer, and multiple bit processor devices	
128	EP 391173 A	☐	Debug peripheral for computers microprocessors, and core processor - provides background mode processing capability in existing CPU core without modification to core design	
129	EP 376004 A	☐	Sequencing of pipe operations - has two modes so that second can be used when first reaches null stage	
130	EP 358293 A	☐	Local area communication transport system - uses broadcasts of services, slot numbers and message sequence numbers to improve transmission of information	
131	EP 349726 A	☐	Integrated-circuit card with low power consumption mode - exchanges information with external data processing system, and has microprocessor	
132	EP 312183 A	☐	Function capability enhancement in data processing system - provides sixteen-bit and thirty-two bit capabilities to system designed originally for eight-bit capability	
133	EP 301707 A	☐	Data processing system using micro-code techniques - has extended processor instruction code mode that disables instructions to interrupt execution of normal instructions	
134	GB 2200482 A	☐	Monitoring control flow in microprocessor - generating representative interface signal when branch instruction is executed	
135	GB 2192471 A	☐	Programmable machine system for food mfr. - has computer controlled actuators and compiler program display to enable operator to enter instructions via input	
136	SU 1352497 A	☐	Data exchange unit - analyses attributes for mode control stored in commands register in control and reception modes	
137	EP 196736 A	☐	Microprogram controlled data processing appts. - has made selector to enable operation of faster and slow microprogram sequences of instructions	
138	DE 3610433 A	☐	Programmed control system for controlling machine - uses stored structural program to fix sequence of contact plan programs	
139	EP 180476 A	☐	Single chip micro-programme sequence controller - can be selectively operated in either interrupt or trapped mode	
140	EP 167333 A	☐	Digital data processor with data word type classifier - provides respective abbreviated jump field in instruction words to specify branch destinations	

is significantly faster than software schedulers. Efficient partitioning of information stored between the hardware and in the host processor significantly reduces the system cost without imposing restrictions.

The present invention utilizes a hardware accelerator, herein called a "task scheduling accelerator" (TSA), for improving the performance of real-time task scheduling without imposing constraints on the functionality of the system. An efficient partitioning between the TSA and the host processor software allows functional flexibility while providing acceleration of task scheduling functionality.

The TSA implements a model of task scheduling where tasks are prioritized and lower priority tasks may be preempted if a higher priority task becomes ready. The TSA may execute commands to add a task, exit the current task, yield to another task at the same priority, modify the current task priority, pend for semaphores or post a semaphore. Using these basic functions, real-time operating system functionality may be developed.

The TSA accepts commands from tasks called "synchronous" task commands. These commands are synchronous from the point of view that the task may not issue another synchronous task command until the previous synchronous task command has been completed. It is allowable for the task to perform other activity after issuing a synchronous task command as long as the task verifies the previous synchronous task command has been completed prior to issuing another synchronous task command. The TSA provides the Task Command register to indicate when a synchronous task command has been recognized and clears the status bit when the function is completed.

Interrupts perform functions on the host processor asynchronous to task execution. Since interrupts may generate events by posting semaphores, it is necessary to have a separate mechanism for issuing commands that are asynchronous to tasks. When these "asynchronous" commands are issued, there is no need to wait for them to complete. An important aspect of a system using the TSA is that the TSA may request service from the host processor via an interrupt. This allows the host processor to additionally service TSA needs or to perform a task switch at the time of the interrupt. If the TSA determines the current task needs to be preempted, the TSA asserts its interrupt. This interrupt causes the host processor to enter a special mode for processing the interrupt request. At this time, the currently executing task has been preempted and a new task can be selected simply by changing to a new context. A task switch is performed by saving the context of the current task and loading the context of a new task. The context is defined as all the registers used by the task and any shared resources. Thus, a task switch involves saving all processor registers and shared resources used by the currently executing task and loading a new set of processor registers and shared resources for the new task.

A task command may cause the current task to be preempted. An example of when this occurs is when a task pends for a semaphore, but the semaphore count is zero or negative. The TSA preempts the currently executing task with an interrupt and switches to another task. Eventually, the task that was preempted becomes ready to execute. When the task continues executing, it determines the TSA status bit is clear and continues. The task is completely unaware preemption has taken place and other tasks were executed. This transparent switching of tasks is important to minimize exposure of the task scheduling functionality to the application program.

In a multi-tasking system, tasks are switched by changing the processor "context". Context is the current information representing the execution state of the task. This typically consists of all of the processor's registers and any additional globally shared resources. The TSA is considered to be a globally shared resource because each task writes task commands to the TSA without regard to other tasks. This introduces a difficult problem with external task scheduling hardware that the present invention solves very effectively. The problem is how the host processor can determine if the TSA registers represent context or not.

This issue is illustrated by an example. Consider the case when an interrupt posts a semaphore that readies a higher priority task. The higher priority task preempts the currently executing task. It is possible that the currently executing task was in the process of writing a task command to the TSA. In this case, this task command should be saved and later restored when the task resumes execution. This allows the task to execute the command it issued just prior to being preempted. In this situation, the TSA registers are considered to be context similar to the host processor's registers.

A second case is illustrated when a task issues a task command that causes the task to be preempted. In this situation, if the TSA's registers are saved and then restored, the command that caused the task switch would be restored causing another task switch. In this situation, the TSA registers should not be considered context. Further, the status bit that indicates the task command was completed needs to be cleared to indicate to the task that the command has completed.

The solution to the TSA "context" is to coordinate the context save/restore procedure between the TSA and the host processor interrupt service routine. This coordination adjusts the context (TSA registers) if the preemption is due to a synchronous task command but preserves the context when the task is being preempted due to an asynchronous task command. The details of this procedure are illustrated in finer detail later.

Another issue with external task scheduling devices is the host processor generating events faster than the TSA can process them. This typically happens when interrupt service routines generate interrupts and post semaphores in a non-predictable fashion. One solution is not to allow interrupts to post semaphores. A system of this type is very seriously constrained because activity from hardware devices cannot be efficiently synchronized. Another potential solution is to have the host processor wait until the TSA can accept additional semaphore posts. This is also an unacceptable solution because it makes interrupt service routines wait and is typically not desirable or allowable.

The solution to this problem is to place additional semaphore posts in a queue. The queuing must be efficient and allow the TSA to control the reading of the queue. The present invention employs a unique solution to this problem by partitioning the storage of additional semaphore posts on the host processor while maintaining the pointers to the queue in the TSA. This combination allows the TSA to be always aware of the state of the host processor queue without interrogating the host processor. Since the size of the queue depends on the software architecture, it is effective to store the queue in the host processor where the queue size can be determined based on the host processor software. The details of this procedure are described in finer detail later.

One embodiment of the present invention is shown in FIG. 1, numeral **100**. FIG. 1 shows a block diagram of a Host Processor **1** coupled to a task scheduling accelerator (TSA)

	Document ID	U	Title	Current OR
141	EP 167241 A	☐	Microprogram sequence controller with comparator register - has comparator which stores instruction address for comparison with generated instruction address on output data bus	
142	EP 150535 A	☐	Loop instruction data processor - uses counter, address register, program counter, clock, ROM, instruction register and loop counter	
143	US 4521850 A	☐	Instruction buffer associated with cache memory unit - stores data group sequence for current procedure while simultaneously storing instruction sequence identified by transfer command	
144	AU 8432301 A	☐	Multi-mode scanning appts. for host and secondary computers - executes sequence of modes by having address of ram scanned into register, and has data transfer between locations permitted using interface	
145	EP 132481 A	☐	Elastic buffer memory for digital multiplex data transmission - indicates RAM address sequence for active channels and extracts micro-instructions at clock rate greater than arrival rate	
146	EP 111776 A	☐	Interrupt controller for instruction pipelined digital processor - has instruction clarification system with interrupt class decoder for each machine instruction and has interrupt sequence detector	
147	EP 126774 A	☐	Microcomputer control system for two-bed desiccant dryer - receives moisture indication from capacitance probes and adjusts regeneration cycle accordingly	
148	DE 3241412 A	☐	Testing advanced direct memory access controller - by using CPU or externally accessing integral test register through sequencer	
149	US 4440059 A	☐	Sound responsive lighting device with VCO driven indexing - sequentially activates each of LEDs, connected to indexer, in which execution speed is proportional to intensity of sound field	
150	US 4430706 A	☐	Branch prediction for data processor - monitors instruction flow and provides record in prediction memory which is accessed using hash coded version	
151	EP 98970 A	☐	Numerical controller for grinding machine - controls machining of cylindrical portions formed on rotating workpiece and has two feed devices responsive to pulse generator	
152	EP 59952 A	☐	Motor vehicle mounted controller - has vocal enunciator to enable occupant to select radio-stereo unit and air conditioning options	
153	US 4342082 A	☐	Program instruction mechanism for shortened recursive handling - sets condition code distinguishing between pending and non-pending request for interruption	
154	EP 35647 A	☐	SIMD parallel array data processor - performs for registration of processing and machine status and activity masks to control execution status	
155	FR 2448123 A	☐	Microprocessor controls rotation sequence of pistol targets - using sequences stored in replaceable memory which may easily be changed	
156	SU 765809 B	☐	Self-diagnosing microprogram processor - has decoder at output of AND=gate with inputs from error-identifying flip=flop and micro-command counter	
157	DE 2911298 B	☐	PCM digital transmission system fault location - using long block words of zeros and loop back line testing through regenerators	

2 that is typically implemented in hardware. The TSA contains registers for control/status or to specify a particular integer value. The control/status registers are shown in FIG. 3, numeral 300.

The Task Command register 301 indicates synchronous task commands issued by tasks on the host processor. These bits include:

Add Task—This bit is set to add a task to the current task set.

Task Exit—This bit is set to exit the currently executing task.

Modify Priority—This bit is set when the currently executing task changes its priority.

Task Yield—This bit is set to yield to another task at the same priority level.

Semaphore Pend—This bit is set when a task makes a pend request.

Task Lock—This bit specifies a mode of operation where the TSA will not preempt the current task. This is typically used to protect critical regions of programs that share data between multiple tasks. When the bit is cleared, the TSA is allowed to preempt the task.

These bits are described in further detail when covering the functions that use them.

The Interrupt Control register 302 specifies the reason for the TSA interrupt to the host processor. There are three major functions this register controls:

Move Post—This is an indication the TSA-ISR is to move a semaphore post index from the semaphore index queue to the asynchronous task command interface. There is no change in the currently executing task.

Move Task—This is an indication to move a task from one task list to another. There is no change in the currently executing task.

Task switch—The task switch is qualified by four bits:

Busy—Indicates the TSA is searching for a ready task.

Save Context—Indicates if the context of the current task should be saved.

Load Context—Indicates if the context of a new task should be loaded.

Idle—Indicates the task set is idle.

These bits are described in further detail when covering the functions that use them.

The Status register 303 specifies additional status information to the host processor or the TSA. These bits include:

CTX—This bit indicates if the TSA-ISR should treat the Task Command register as context (preserve it) or not as context (modify it). This bit is used to clear command bits in the Task Command register during a context switch.

Semaphore Post—This bit indicates a task or interrupt service routine has written a semaphore post to the asynchronous task command interface.

IDLE—This bit is used to retain the state of the task set being in the idle (no active tasks) state.

Preemption Pending—This bit is set to indicate the current task will be preempted when it turns off the task lock. This bit is set by the TSA if a semaphore is posted with a higher priority ready task pending when the task lock is enabled.

These bits are described in further detail when covering the functions that use them.

When the system is reset, the TSA begins execution of its internal State Machine 41 as indicated in FIG. 4, numeral 400. At initialization 4111, the State Machine 41 clears:

The Current Execution Priority 44. The Current Execution Priority 44 maintains the priority of the currently executing task. In this embodiment, a zero indicates the highest priority with lower priorities increasing in value. In this embodiment flowcharts indicate "higher priority" to mean a lower numeric value and "lower priority" to indicate a higher numeric value for priority. Since the Current Execution Priority 44 is being initialized to zero, it indicates the first task created by starting the host processor is currently the highest priority.

The Ready Task Counters 43. The Ready Task Counters store a plurality of integers that determines the number of ready tasks at each priority level. Since at reset there are no other ready tasks to execute, all Ready Task Counters 43 are set to zero.

All flags in the Status register 42. This includes

The CTX Flag. This bit indicates if the Synchronous Task Command Interface (STCI) 46 contains context. If the CTX bit is set, it indicates the STCI registers should be preserved because they may contain valid information written by the task that is currently executing. If the CTX bit is clear, it indicates the STCI registers have issued a task command that caused the context switch and the STCI registers should be modified to clear the task command status bit. It is initialized to zero by default.

The Semaphore Post flag. This bit indicates if the semaphore post registers in the Asynchronous Task Command Interface 45 have been written.

The IDLE flag. This bit indicates the host processor is currently idle and not processing any tasks.

The Preemption Pending flag. This bit indicates a ready task of higher priority will preempt the current task when the task lock is turned off.

The Interrupt Control register 403 in the Interrupt Service Routine Interface 40. This register is cleared because if any bit in the Interrupt Control register 403 is set, Interrupt Generator 404 asserts the interrupt 30. Since the TSA does not need any service at start-up, it is required to ensure the interrupt is negated after reset.

The Task Command register 465 in the Synchronous Task Command Interface 46. This register is cleared to indicate there are currently no task commands to process.

After initialization 4111, the State Machine 41 enters a loop checking for a function to execute with comparisons 4120, 4130, 4140, 4150, 4160, 4190 and 4170. The loop checking includes: Is Sem Pend set in the TCR (Task Command Register)? 4120; Is Sem Post set in Status? 4130; Is Add Task set in TCR? 4140; Is Task Exit set in TCR 4150; Is Task Yield set in TCR? 4160; Is Modify Priority set in TCR? 4190; Does S/W queue have sem posts and are post registers empty? 4170; and, Is the task lock off and the preemption pending bit set? 4195. If any comparison is true, the TSA processes the true function 4122, 4132, 4142, 4152, 4162, 4192, 4172, 4196 and returns to the beginning of the loop to check Is Sem Pend set in the TCR 4120. If all comparisons are false, the TSA does not have any functions to process and enters a low power mode 4180 until there is activity. Each of the functions of the State Machine 41 are explained in further detail later. Activity that brings the State Machine 41 out of low power 4180 is any of the loop conditions being true. Since the state machine is turned off when in low power, activity is detected by direct hard-wired logic to the state machine. This is shown in detail in FIG. 32 in the Clock Activate Logic 801600 where signals Sema-